

CSCI114

Procedural Programming

Class Examples

Year 2017

Dr Heng Aik Koan

All examples developed in class should be read in conjunction with the lecture notes provided by UOW. Please do not pass these class examples and all your works to your juniors so that they can concentrate and can learn in class. Their futures depend on you. Guide them in the right way of learning



All rights *not* reserved



Topics you should now in CSCI114

Table of contents (Class examples Only)

1. Internal representation
 - How to convert numbers to binary, octal and hexadecimal?
- How to convert numbers from binary, octal, hexadecimal to decimals?
 - How to convert from octal to hexadecimal and vice versa?
 - Representation of negative number, 1s or 2s representation.
 - Arithmetic (+ and -) on different bases.
2. sequential design
 - basic computing
 - how to declare and use variables?
 - cin & cout functions
3. selection design
 - if
 - if else
 - switch case
4. repetition design
 - while loop
 - do-while loop
 - for loop
5. Function I
 - passing by values
6. Function II
 - passing by reference
 - default parameters
 - storage class
 - recursive function
 - pseudo random numbers
 - bitwise manipulation
7. Arrays
 - One Dimensional arrays
- Two dimensional arrays
- C-strings
- Functions in cstring library
- Functions in ctype library

8. User defined data types
 - Enumeration data types
 - Structures
9. File processing
 - Relate file processing with array of structures
10. Sample questions

Representation of numbers

1. Convert a decimal number to binary.

For example, $n = 197$

$$197 / 2 = 98 \text{ R } 1$$

$$98 / 2 = 49 \text{ R } 0$$

$$49 / 2 = 24 \text{ R } 1$$

$$24 / 2 = 12 \text{ R } 0$$

$$12 / 2 = 6 \text{ R } 0$$

$$6 / 2 = 3 \text{ R } 0$$

$$3 / 2 = 1 \text{ R } 1$$

$$1 / 2 = 0 \text{ R } 1$$

copy the remainders from bottom to top

11000101 (binary representation of 197)

Working on 24 bits representation, we have

00000000000000000011000101

2. Convert a decimal number to octal

For example, $n = 197$ to octal

$$197 / 8 = 24 \text{ R } 5$$

$$24 / 8 = 3 \text{ R } 0$$

$$3 / 8 = 0 \text{ R } 3$$

Octal representation of 197 is

305

3. Convert an octal number to decimal.

For example, octal number 305 to decimal

305 (in octal)

$$= 3 * 8 * 8 + 0 * 8 + 5$$

$$= 192 + 5$$

$$= 197 \text{ (decimal)}$$

4. Convert hexadecimal number FB5 to decimal

FB5

$$= F * 16 * 16 + B * 16 + 5$$

$$= 15 * 256 + 11 * 16 + 5$$

= ...

Conclusion: we can convert from any base (2 or 8 or 16) to decimal

Some remarks

Octal Binary

0 000

1 001

2 010

3 011

4 100

5 101

6 110

7 111

3 bits can represent an octal symbol

5. Convert binary number 11000101 to octal equivalent

11000101 (binary)

= 11 000 101 (group 3 by 3 from right to left)

= 3 0 5 (octal)

Other remark

Hexadecimal Binary

0 0000

1 0001

2 0010

3 0011

4 0100

5 0101

6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

6. Given binary number 11101011111001110111, what is its hexadecimal equivalent?

11101011111001110111 (binary)
 = 1110 1011 1110 0111 0111 (4 by 4 from right to left)
 = E B E 7 7 (hexadecimal)

7. From octal to hexadecimal (or vice versa)???

ABCF9D6 (hexadecimal)
 = 1 010 101 111 001 111 100 111 010 110 (binary)
 = 1257174726 (octal)

8. Add two octal numbers 457 and 675. What is the result in hexadecimal?

```

  / /
  4 5 7
  6 7 5 +
  -----
  1 3 5 4 (octal)
  = 0010 1111 1100 (binary)
  = 2FC (hexadecimal)

```

calculations

7 + 5 = 12(decimal) = 14 (octal)
 1 + 5 + 7 = 13 (decimal) = 15 (octal)
 1 + 4 + 6 = 11 (decimal) = 13 (octal)

9. Add two hexadecimal numbers BFC and ABC. What is the result in octal?

```

  / /
  B F C
  A B C +
  -----

```

1 6 B 8 (hexa)
 = 0 001 011 010 111 000 (binary)
 = 1 3 2 7 0 (octal)

Calculations

$C + C = 12 + 12 = 24(\text{decimal}) = 18 \text{ (hexadecimal)}$
 $1 + F + B = 1 + 15 + 11 = 27 \text{ (decimal)} = 1B \text{ (hexadecimal)}$
 $1 + B + A = 1 + 11 = 10 = 22 \text{ (decimal)} = 16 \text{ (hexa)}$

10. Compute hexadecimal expression $BCD - 8FE = ???$

B C D
 8 F E -

 2 C F

Calculation

$16 + D - E = 16 + 13 - 14 = 15 = F$
 $16 + B - F = 16 + 11 - 15 = 12 = C$

Representation of negative numbers

=====

Working on 24 bits representation, what is the 2's complement of 167 in binary? In octal? And in hexadecimal?

step 1: convert 167 to binary

```
167 / 2 = 83 R 1
83 / 2 = 41 R 1
41 / 2 = 20 R 1
20 / 2 = 10 R 0
10 / 2 = 5 R 0
5 / 2 = 2 R 1
2 / 2 = 1 R 0
1 / 2 = 0 R 1
```

Binary representation is
10100111

Internal representation (based on 24 bits)
0000 0000 0000 0000 1010 0111

Step 2: 1's complement
1111 1111 1111 1111 0101 1000

Step 3: 2's complement (+ 1 to 1's)
1111 1111 1111 1111 0101 1001
F F F F 5 9 (hexa)

111 111 111 111 111 101 011 001
7 7 7 7 7 5 3 1 (octal)

// The use of / and % operators

```
#include <iostream>
using namespace std;

int main ()
{
    int m;

    cout << "Enter a positive integer: ";
    cin >> m;

    cout << "The last digit of " << m
         << " is "
         << m % 10
         << endl;

    // Eliminate the last digit of m
    m = m / 10;

    cout << "After elimination of last digit" << endl;
    cout << "We have " << m << endl;
}
```

// How to swap two values?

```
#include <iostream>
using namespace std;

int main ()
{
    int m = 888;
    int n = 999;
    int temp;

    temp = m;
    m = n;
    n = temp;

    cout << m << " - " << n << endl;
}
```

```

// read in a radius of a circle
// and compute the area and the
// circumference
//
// File name: circle.cpp
#include <iostream>
using namespace std;

const double PI = 3.14159256;

int main ()
{
    int radius;
    double area, perimeter;

    cout << "Enter a radius: ";
    cin >> radius;

    // compute the area and perimeter
    area = PI * radius * radius;
    perimeter = 2.0 * PI * radius;

    // Display the results
    cout << "Area = "
         << area
         << endl;

    cout << "Perimeter = "
         << perimeter
         << endl;

    return 0;
}

```

```

// Read in the length and the width of
// two sets of rectangles, compute their
// areas and their perimeters and display
// the result according to the following
// patterns:
//
// Length Width Area Perimeter
//
// ***      ***      ***      ***
// ***      ***      ***      ***
#include <iostream>
using namespace std;

int main ()
{
    int length1, length2;
    int width1, width2;
    int area1, area2;
    int perimeter1, perimeter2;

    cout << "Enter length 1 and width 1: ";
    cin >> length1 >> width1;

    cout << "Enter length 2 and width 2: ";
    cin >> length2 >> width2;

    // Compute area and perimeter
    area1 = length1 * width1;
    area2 = length2 * width2;
    perimeter1 = 2 * (length1 + width1);
    perimeter2 = 2 * (length2 + width2);

    // Display the results
    cout << "Length   Width   Area   Perimeter" << endl;
    cout << endl;

    cout << length1 << "   "
        << width1 << "   "
        << area1 << "   "
        << perimeter1 << "   "
        << endl;

    cout << length2 << "   "
        << width2 << "   "

```

```
        << area2 << "    "  
        << perimeter2 << "    "  
        << endl;  
  
    return 0;  
}
```

```
// A c++ program to convert a degree from
// celsius to fahrenheit
#include <iostream>
using namespace std;

int main ()
{
    float degreeC, degreeF;

    cout << "Enter a degree in C: ";
    cin >> degreeC;

    degreeF = (9.0 / 5) * degreeC + 32;

    cout << "Degree in F = "
         << degreeF
         << endl;

    return 0;
}
```

```
#include <iostream>
using namespace std;

int main()
{
    int num1 = 1, num2 = 2;

    cin >> num1 >> num2;
    cout << num1 << " " << num2 << endl;

    // clear and ignore the next 100 characters
    // or the end of line is hit, whichever
    // comes first
    cin.clear ();
    cin.ignore (100, '\n');

    cin >> num1 >> num2;
    cout << num1 << " " << num2 << endl;
    return 0;
}
```

```

// Formatted outputs
// Read in the length and the width of two rectangles
// and display their area and perimeters according
// to the following format
//
//      Length      Width      Area      Perimeter
// -----
//      xxx        xxxx       xxxxx      xxxx
//      xxx        xxx        xxx        xx
// =====
//
#include <iostream>
#include <iomanip>
using namespace std;

int main ()
{
    int length1, width1;
    int length2, width2;

    cout << "Enter length and width: ";
    cin >> length1 >> width1;

    cin.clear ();
    cin.ignore (100, '\n');

    cout << "Enter length and width: ";
    cin >> length2 >> width2;

    cout << endl;

    int area1 = length1 * width1;
    int area2 = length2 * width2;
    int perimeter1 = 2 * (length1 + width1);
    int perimeter2 = 2 * (length2 + width2);

    cout << left << setw (10) << "Length"
         << left << setw (10) << "Width"
         << left << setw (10) << "Area"
         << left << setw (10) << "Perimeter"
         << endl;

    cout << setw (40) << setfill ('-') << "-" << endl;

```

```
cout << setfill (' ');

cout << setw (10) << length1
    << setw (10) << width1
    << setw (10) << area1
    << setw (10) << perimeter1
    << endl;

cout << setw (10) << length2
    << setw (10) << width2
    << setw (10) << area2
    << setw (10) << perimeter2
    << endl;

cout << setw (40) << setfill ('=') << "=" << endl;
cout << setfill (' ');
}
```

```

// Formatted outputs
// Read in the length and the width of two rectangles
// and display their area and perimeters according
// to the following format
//
//      Length      Width      Area      Perimeter
// -----
//      xxx        xxxx       xxxxx      xxxx
//      xxx        xxx        xxx        xx
// =====
//
// Note the use of left and right justified command

```

```

#include <iostream>
#include <iomanip>
using namespace std;

int main ()
{
    float length1, width1;
    float length2, width2;

    cout << "Enter length and width: ";
    cin >> length1 >> width1;

    cin.clear ();
    cin.ignore (100, '\n');

    cout << "Enter length and width: ";
    cin >> length2 >> width2;

    cout << endl;

    float area1 = length1 * width1;
    float area2 = length2 * width2;
    float perimeter1 = 2 * (length1 + width1);
    float perimeter2 = 2 * (length2 + width2);

    cout << fixed << showpoint << setprecision (4);

    cout << left << setw (10) << "Length"
         << left << setw (10) << "Width"
         << left << setw (10) << "Area"

```

```

        << left << setw (10) << "Perimeter"
        << endl;

cout << setw (40) << setfill ('-') << "-" << endl;
cout << setfill (' ');

cout << right << setw (10) << length1
    << right << setw (10) << width1
    << right << setw (10) << area1
    << right << setw (10) << perimeter1
    << endl;

cout << right << setw (10) << length2
    << right << setw (10) << width2
    << right << setw (10) << area2
    << right << setw (10) << perimeter2
    << endl;

cout << setw (40) << setfill ('=') << "=" << endl;
cout << setfill (' ');
}

```

What is the result of the hexadecimal expression DE – AB in octal?

- (A) 63
- (B) 55
- (C) 33
- (D) 35
- (E) None of the above

Calculation

$$\begin{array}{r} \text{D E} \\ \text{A B -} \\ \hline \end{array}$$

3 3 (hexadecimal)
= 00 110 011 (binary)
= 6 3 (octal)

What is the result of the hexadecimal expression EC – CA in decimal?

- (A) 22
- (B) 34
- (C) 42
- (D) 44
- (E) None of the above

$$\begin{array}{r} \text{E C} \\ \text{C A -} \\ \hline \end{array}$$

2 2 (hexadecimal)
= 2 * 16 + 2
= 34 (decimal)

Assignment operators

+=
-=
*=
/=

%=

For example

```
int x = 8;  
int y = 9;  
int z = 10;
```

```
x += y - z;  
==> x = x + (y - z)  
     = 8 + (9 - 10)  
     = 7
```

```
x *= x + y - z;  
==> x = x * (x + y - z)  
     = 8 * (8 + 9 - 10)  
     = 8 * 11  
     = 88
```

Suppose x, y, z are int variables and x=2, y=3, z=1.
What is the output of following program fragment?

```
x += 5 + y;  
y *= 2 + x - z;  
z = y--;  
cout << x << " " << y << " " << z;
```

- (A) 10 15 14
- (B) 10 11 11
- (C) 10 14 14
- (D) 10 14 15
- (E) None of the above

Calculations

$$x = x + 5 + y = 2 + 5 + 3 = 10$$

$$\begin{aligned} y &= y * (2 + x - z) \\ &= 3 * (2 + 10 - 1) \\ &= 3 * 11 \\ &= 33 \end{aligned}$$

```
z = y--  
==> z = 33  
    y = 32
```

Truth table in mathematics

A	B	A && B	A B	~A
T	T	T	T	F
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

```

#include <iostream>
#include <iomanip>
using namespace std;

int main ()
{
    int m, n;
    char ch1, ch2;
    float x, y;

    cout << fixed << showpoint << setprecision (1);

    cin >> ch1 >> m >> x;
    cin.get (ch2);

    cin.clear ();
    cin.ignore (100, '\n');

    cin >> n >> y;

    cout << m << x << ch1 << ch2 << y << n << endl;
}

```

/* We enter the following input

```

123.23.ABCD
12.456ABCD
*/

```

```

#include <iostream>
#include <iomanip>
using namespace std;

int main ()
{
    int m = -5, n = -3;

    m -= m - n * 3;

    /*
        m = m - (m - n * 3)
          = -5 - (-5 + 3 * 3)
          = -5 - (-5 + 9)
          = -5 - (4)
          = -9

        n = -4;
    */

    --n;

    cout << m << n << endl;

    int k = ++5 - -6;
    cout << k << endl; // 11
}

```

```

// an example on a simple if statement
// - read in a character and see this character
// is an alphabet. If this is an alphabet is it
// an uppercase letter
#include <iostream>
using namespace std;

int main ()
{
    char ch;

    cout << "Enter a character: ";
    cin.get (ch);

    if ((ch >= 'A' && ch <= 'Z') ||
        (ch >= 'a' && ch <= 'z'))
    {
        cout << ch << " is alphabet" << endl;

        if (ch >= 'A' && ch <= 'Z')
            cout << ch << " is an uppercase letter" << endl;
    }
}

```

// A test of odd and even

```
#include <iostream>
using namespace std;

int main ()
{
    int n;

    cout << "Enter an integer: ";
    cin >> n;

    if (n % 2)
        cout << n << " is odd" << endl;
    else
        cout << n << " is even" << endl;
}
```

```
// an example on a simple if statement  
// - read in a character and see this character  
// is an alphabet. If this is an alphabet is it  
// an uppercase letter or a lowercase letter
```

```
#include <iostream>  
using namespace std;
```

```
int main ()  
{  
    char ch;  
  
    cout << "Enter a character: ";  
    cin.get (ch);  
  
    if ((ch >= 'A' && ch <= 'Z') ||  
        (ch >= 'a' && ch <= 'z'))  
    {  
        cout << ch << " is alphabet" << endl;  
  
        if (ch >= 'A' && ch <= 'Z')  
            cout << ch << " is an uppercase letter" << endl;  
        else  
            cout << ch << " is a lowercase letter" << endl;  
    }  
    else  
        cout << ch << " is not an alphabet" << endl;  
}
```

// an example on a simple if statement
// - read in a character and see this character
// is an alphabet. If this is an alphabet is it
// an uppercase letter or a lowercase letter
// If it is a digit, is it odd or even?
// If it is a special character is it a space?

```
#include <iostream>
using namespace std;

int main ()
{
    char ch;

    cout << "Enter a character: ";
    cin.get (ch);

    if ((ch >= 'A' && ch <= 'Z') ||
        (ch >= 'a' && ch <= 'z'))
    {
        cout << ch << " is alphabet" << endl;

        if (ch >= 'A' && ch <= 'Z')
            cout << ch << " is an uppercase letter" << endl;
        else
            cout << ch << " is a lowercase letter" << endl;
    }
    else if (ch >= '0' && ch <= '9')
    {
        cout << ch << " is a digit" << endl;

        int n = ch - '0';

        if (n % 2 == 0)
            cout << ch << " is an even digit" << endl;
        else
            cout << ch << " is an odd digit" << endl;
    }
    else if (ch == ' ')
        cout << "\"" << ch << "\"" << " is a space" << endl;
    else
        cout << ch << " is a nonspace special symbol" << endl;
    }

    /*
```

'0' '1' '2' '3' '4' '5' '6' '7' '8' '9'

'0' - '0' = 0

'1' - '0' = 1

'2' - '0' = 2

etc

*/

// an example on a simple if statement
// - read in a character and see this character
// is an alphabet. If this is an alphabet is it
// an uppercase letter or a lowercase letter
// If it is a digit, is it odd or even?
// If it is a special character is it a space?
// use of switch case statement

```
#include <iostream>
using namespace std;

int main ()
{
    char ch;

    cout << "Enter a character: ";
    cin.get (ch);

    if ((ch >= 'A' && ch <= 'Z') ||
        (ch >= 'a' && ch <= 'z'))
    {
        cout << ch << " is alphabet" << endl;

        if (ch >= 'A' && ch <= 'Z')
            cout << ch << " is an uppercase letter" << endl;
        else
            cout << ch << " is a lowercase letter" << endl;
    }
    else if (ch >= '0' && ch <= '9')
    {
        cout << ch << " is a digit" << endl;

        switch (ch)
        {
            case '0':
            case '2':
            case '4':
            case '6':
            case '8': cout << ch << " is an even digit" << endl;
                    break;
            default : cout << ch << " is an odd digit" << endl;
                    break;
        }
    }
    else if (ch == ' ')

```

```
        cout << "\"" << ch << "\"" << " is a space" << endl;
    else
        cout << ch << " is a nonspace special symbol" << endl;
}
```

**// Define a few constans to study the propeties
// of a character.**

// if the charcter is letters, is it a vowel?

// if the character is digits, is it odd or even?

// if the character is others, is is a space or

// non space symbols?

#include <iostream>

using namespace std;

const char LETTER = 'L';

const char DIGIT = 'D';

const char SPACE = ' ';

const char SPECIAL = 'S';

int main ()

{

 char ch;

 char type;

 cout << "Enter a character: ";

 cin.get (ch);

 if ((ch >= 'A' && ch <= 'Z') ||

 (ch >= 'a' && ch <= 'z'))

 type = LETTER;

 else if (ch >= '0' && ch <= '9')

 type = DIGIT;

 else

 type = SPECIAL;

 switch (type)

 {

 case LETTER: cout << ch << " is an alphabet" << endl;

 switch (ch)

 {

 case 'a':

 case 'A':

 case 'e':

 case 'E':

 case 'i':

 case 'I':

 case 'o':

 case 'O':

 case 'u':

```

        case 'U': cout << "It is a vowel"
<< endl;
        break;
        default : cout << "It is not a vowel"
        << endl;
    }
    break;
    case DIGIT: cout << ch << " is a digit" << endl;
    switch (ch)
    {
        case '0':
        case '2':
        case '4':
        case '6':
        case '8': cout << "It is even digit"
<< endl;
        break;
        default : cout << "It is odd digit"
        << endl;
    }
    break;
    default:  cout << "\"
    <<ch << "\"
    << " is a special symbol"
    << endl;

    switch (ch)
    {
        case SPACE: cout << "It is a space" << endl;
        break;
        default: cout << "It is nonspace symbol"
        << endl;
    }
}
}

```

**// Define a few constans to study the propeties
of a character.
// if the charcter is letters, is it a vowel?
// if the character is digits, is it odd or even?
// if the character is others, is is a space or
// non space symbols?**

```

#include <iostream>
using namespace std;

```

```

const char LETTER = 'L';
const char DIGIT = 'D';
const char SPACE = ' ';
const char SPECIAL = 'S';

int main ()
{
    char ch;
    char type;

    cout << "Enter a character: ";
    cin.get (ch);

    if ((ch >= 'A' && ch <= 'Z') ||
        (ch >= 'a' && ch <= 'z'))
        type = LETTER;
    else if (ch >= '0' && ch <= '9')
        type = DIGIT;
    else
        type = SPECIAL;

    switch (type)
    {
        case LETTER: cout << ch << " is an alphabet" << endl;
                    if (ch == 'a' || ch == 'A' ||
                        ch == 'i' || ch == 'I' ||
                        ch == 'e' || ch == 'E' ||
                        ch == 'u' || ch == 'U' ||
                        ch == 'o' || ch == 'O')
                        cout << ch << " is a vowel" << endl;
                    else
                        cout << ch << " is not a vowel"
<< endl;
                    break;
        case DIGIT: cout << ch << " is a digit" << endl;
                    switch (ch)
                    {
                        case '0':
                        case '2':
                        case '4':
                        case '6':
                        case '8': cout << "It is even digit" << endl;

```

```

        break;
        default : cout << "It is odd digit" << endl;
    }
    break;
default: cout << "\"
        <<ch << "\"
        << " is a special symbol"
        << endl;

    switch (ch)
    {
        case SPACE: cout << "It is a space" << endl;
                    break;
        default: cout << "It is nonspace symbol" << endl;
    }
}
}

```

// convert the following to switch statement

```
int percentOff;
char type;

cout << "Input type: ";
cin >> type;
if (type == 'c')
    percentOff = 50;
else if (type == 't')
    percentOff = 75;
else if (type == 'p' || type == 'q')
    percentOff = 20;
else if (type == 'v')
    percentOff = 30;
else if (type == 'f')
    percentOff = 10;
else
    percentOff = 5;
```

```
switch (type)
{
    case 'c': percentOff = 50;
               break;
    case 't': percentOff = 75;
               break;
    case 'p':
    case 'q': percentOff = 30;
               break;
    case 'f': percentOff = 10;
               break;
    default : percentOff = 5;
}
```

```

int sum = 0;
while ( sum <= 10)
{
    sum = sum + 2;
    cout << sum << " ";
}

```

sum	sum <= 10?	out << sum << " "
0	0 <= 10? yes	2 4 6 8 10 12
2	2 <= 10? yes	
4	4 <= 10? yes	
6	6 <= 10? yes	
8	8 <= 10? yes	
10	10 <= 10? yes	
12	12 <= 10? no	

```

// Design a program to find the sum of
// digits in a positive integer n.
// For example, n = 12345
// sum of digits = 1 + 2 + 3 + 4 + 5
//                      = 15
//                      = 5 + 4 + 3 + 2 + 1

```

// Algorithm:

```

// int sum = 0
// while n is positive
//     add the last digit to the sum
//     remove the last digit
//
//     print out the sum

```

```

#include <iostream>
using namespace std;

```

```

int main ()
{
    int n, lastDigit;
    int sum;

    cout << "Enter a positive integer: ";
    cin >> n;

    sum = 0;
    while (n > 0)
    {
        lastDigit = n % 10;
        sum = sum + lastDigit;

        n = n / 10;
    }

    cout << "The sum of digits = "
         << sum
         << endl;
}

```

```

// Design a program to find the sum of
// all the odd digits and the sum of all
// the even digits occur in a positive integer n.

```

```

// For example, n = 12345
// sum of odd digits = 9
// sum of even digits = 6

// int sumOdds = 0
// int sumEvens = 0;

// while n is positive
//         if the last digit is odd,
//             add to sumOdds
//         else
//             add to sumEvens
//         remove the last digit
//
//     print out the sum

#include <iostream>
using namespace std;

int main ()
{
    int n, lastDigit;
    int sumOdds, sumEvens;

    cout << "Enter a positive integer: ";
    cin >> n;

    sumOdds = 0;
    sumEvens = 0;

    while (n > 0)
    {
        lastDigit = n % 10;

        if (lastDigit % 2)
            sumOdds += lastDigit;
        else
            sumEvens += lastDigit;

        n = n / 10;
    }

    cout << "The sum of odd digits = "

```

```
        << sumOdds
        << endl;

cout << "The sum of even digits = "
    << sumEvens
    << endl;

}
```

```
// Design a program to find the sum of
// all the odd digits and the sum of all
// the even digits occur in a positive integer n.
```

```
// For example, n = 12345
// sum of odd digits = 9
// sum of even digits = 6
```

```
// int sumOdds = 0
// int sumEvens = 0;
```

```
// while n is positive
//     if the last digit is odd,
//         add to sumOdds
//     else
//         add to sumEvens
//     remove the last digit
//
// print out the sum
```

```
#include <iostream>
using namespace std;
```

```
int main ()
{
    int n, lastDigit;
    int sumOdds, sumEvens;

    cout << "Enter a positive integer: ";
    cin >> n;

    sumOdds = 0;
    sumEvens = 0;

    while (n > 0)
    {
        lastDigit = n % 10;

        switch (lastDigit % 2)
        {
            case 1: sumOdds += lastDigit;
                    break;
            default: sumEvens += lastDigit;
```

```
    }  
    n = n / 10;  
}  
  
cout << "The sum of odd digits = "  
    << sumOdds  
    << endl;  
  
cout << "The sum of even digits = "  
    << sumEvens  
    << endl;  
  
}
```

**// Design a program to find the sum of
// all the odd digits and the sum of all
// the even digits occur in a positive integer n.**

**// For example, n = 12345
// sum of odd digits = 9
// sum of even digits = 6**

**// int sumOdds = 0
// int sumEvens = 0;**

**// while n is positive
// if the last digit is odd,
// add to sumOdds
// else
// add to sumEvens
// remove the last digit
//
// print out the sum**

#include <iostream>
using namespace std;

```
int main ()
{
    int n, lastDigit;
    int sumOdds, sumEvens;

    cout << "Enter a positive integer: ";
    cin >> n;

    sumOdds = 0;
    sumEvens = 0;

    while (n > 0)
    {
        lastDigit = n % 10;

        sumOdds += (lastDigit % 2? lastDigit : 0);
        sumEvens += (lastDigit % 2? 0 : lastDigit);

        n = n / 10;
    }
}
```

```
cout << "The sum of odd digits = "  
    << sumOdds  
    << endl;  
  
cout << "The sum of even digits = "  
    << sumEvens  
    << endl;  
  
}
```

```
int num = 1;

while ( num <= 10)
{
    cout << num << " ";
    num = num * 2;
}
```

num	num <= 10	cout << ... << " "
1	1 <= 10? y	1 2 4 8
2	2 <= 10? y	
4	4 <= 10? Y	
8	8 <= 10? Y	
16	16 <= 10? N	

```
int count = 1;    // initialise count
```

```
while (count <= 7)
{
    cout << count << " ";
    count++;
}
```

count	count <= 7	cout << ... << " "
1	1 <= 7? y	1 2 3 4 5 6 7
2	2 <= 7? y	
3	3 <= 7? y	
...		
7	7 <= 7? y	
8	8 <= 7? n	

```
int totalMarks = 0, marks = 0;
```

```
while (marks >= 0)
{
    cout << "Enter a mark (-1 to end): ";
    cin >> marks;    // 1 2 3 -1

    totalMarks += (marks >= 0 ? marks : 0);
}
```

```
cout << "Total Marks: " << totalMarks << endl;
```

```
marks marks >= 0? totalMarks (= 0)
```

0	0 >= 0? y	
1		1
	1 >= 0? y	
2		3
	2 >= 0? y	
3		6
	3 >= 0? y	
-1		6

```
int num, result = 1;
```

```
cout << "Enter a number: ";
cin >> num;           // 4
```

```
while (num > 0)
{
    result = result * num--;
}
```

```
cout << "result is: " << result << endl;
```

num	num > 0	result (= 1)
4	4 > 0? y	4
3	3 > 0? y	12
2	2 > 0? y	24
1	1 > 0? y	24
0	0 > 0? n	

```
int num, result = 1;
```

```
cout << "Enter a number: ";
cin >> num; // 4
```

```
while (num > 0)
{
    result = result * --num;
}
```

```
cout << "result is: " << result << endl;
```

num	num > 0?	result
4	4 > 0? y	
3		3
	3 > 0? y	
2		6

```

    2 > 0? y
1
    1 > 0? y
0
    0

```

```
int count;
```

```
for (count = 0; count < 10; count = count + 2)
    cout << count << " ";
```

```
count count < 10  cout << ....
0      0 < 10? Y   0 2 4 6 8
2      2 < 10? y
4      4 < 10? y
6      6 < 10? y
8      8 < 10? y
10     10 < 10? n

```

```
for (...; ...; ...)
{

}

```

```
for ( ; ...; ...)
{

}

```

```
for ( ; ...; )
{

}

```

```
for ( ; ; )
{
    // you need to take a break!!!!
}

```

```
int i, j;
```

```

for (i=0; i < 3; i++)
    for (j=3; j >= 0; j--)
        cout << i << ", " << j << endl;

```

i	j	cout << ...
0	3	0, 3
	2	0, 2
	1	0, 1
	0	0, 0
1	3	1, 3
	2	1, 2
	1	1, 1
	0	1, 0
2	3	2, 3
	2	2, 2
	1	2, 1
	0	2, 0

```

int counter, sum, N;

```

```

cout<<"Enter the number of positive integers to be added:" << endl;
cin >> N;    // 5

```

```

sum = 0;
for(counter = 1; counter <= N; counter++)
    sum = sum + counter;

```

```

cout<<"The sum of the first " << N
    << " positive integers is " << sum << endl;

```

counter	counter <= 5?	sum (= 0)
1	1 <= 5? y	1
2	2 <= 5? y	3
3	3 <= 5? y	6
4	4 <= 5? y	10
5	5 <= 5? y	15
6	6 <= 5? n	

// Flag controlled while loop
// We wish to check if a digit is
// inside a positive integer.
// For example nn 12345, digit 3 is inside
// digit 7 is not

```
#include <iostream>
using namespace std;

int main ()
{
    int n;
    int aDigit;

    cout << "Enter an integer: ";
    cin >> n;

    cout << "Enter a digit: ";
    cin >> aDigit;

    bool found = false;

    while (!found && n > 0)
    {
        if (aDigit == n % 10)
            found = true;
        else
            n /= 10;
    }

    if (found)
        cout << "Digit found" << endl;
    else
        cout << "Digit not found" << endl;

    return 0;
}
```

```

// Flag controlled while loop
// Counter controlled while loop
// Read in a few set of integers + some digits
// For each set:
//     We wish to check if a digit is
//     inside a positive integer.
//     For example n = 12345, digit 3 is inside
//     digit 7 is not

```

```

#include <iostream>
using namespace std;

int main ()
{
    int n;
    int aDigit;

    int limit;

    cout << "How many set: ";
    cin >> limit;

    int counter = 1;

    while (counter <= limit)
    {
        cout << "Enter integer " << counter << ": ";
        cin >> n;

        cout << "Enter a digit: ";
        cin >> aDigit;

        bool found = false;

        while (!found && n > 0)
        {
            if (aDigit == n % 10)
                found = true;
            else
                n /= 10;
        }

        if (found)
            cout << "Digit found" << endl;
    }
}

```

```
        else
            cout << "Digit not found" << endl;

        counter ++;

        cout << endl;
    }
    return 0;
}
```

```

// Flag controlled while loop
// Counter controlled while loop
// Read in a few set of integers (not more than 10)
// + some digits
// For each set:
//     We wish to check if a digit is
//     inside a positive integer.
//     For example n = 12345, digit 3 is inside
//     digit 7 is not

```

```

#include <iostream>
using namespace std;

```

```

int main ()
{
    int n;
    int aDigit;

    int limit;

    // Read with data validation
    do
    {
        cout << "How many set: ";
        cin >> limit;
    } while (limit <= 0 || limit >= 10);

    cout << endl;

    int counter = 1;

    while (counter <= limit)
    {
        cout << "Enter integer " << counter << ": ";
        cin >> n;

        cout << "Enter a digit: ";
        cin >> aDigit;

        bool found = false;

        while (!found && n > 0)
        {
            if (aDigit == n % 10)

```

```

        found = true;
    else
        n /= 10;
}

if (found)
    cout << "Digit found" << endl;
else
    cout << "Digit not found" << endl;

counter ++;

cout << endl;
}
return 0;
}

```

/*

An integer is divisible by 9 if the sum of its digits is divisible by 9. Write a program that prompts the user to input an integer. The program should then output the number and a message stating whether the number is divisible by 9.

*/

```
#include <iostream>
using namespace std;
```

```
int main ()
{
    int n;
    int sumDigits;
    bool ok;

    // Read a positive integer
    do
    {
        cout << "Enter a positive integer: ";
        cin >> n;
    } while (n <= 0);

    // Get the sum of digits
    sumDigits = 0;
    while (n > 0)
    {
        sumDigits = sumDigits + n % 10;
        n = n / 10;
    }

    // Test divisible by 9
    if (sumDigits % 9 == 0)
        ok = true;
    else
        ok = false;

    // print out the result
    if (ok)
        cout << "Divisible by 9" << endl;
    else
        cout << "Not divisible by 9" << endl;
}
```



```
/*
```

An integer is divisible by 9 if the sum of its digits is divisible by 9. Write a program that prompts the user to input an integer. The program should then output the number and a message stating whether the number is divisible by 9.

```
*/
```

```
#include <iostream>
using namespace std;
```

```
// THIS IS FOR DEMONSTRATION ONLY. IN FUTURE
// YOU SHOULD NOT HAVE VARIABLES OUTSIDE
// THE FUNCTIONS
```

```
    int n;
    int sumDigits;
    bool ok;
```

```
void getAnInteger ();
void getSumOfDigits ();
void testDivisibleBy9 ();
void printResult ();
```

```
int main ()
{
    getAnInteger ();
    getSumOfDigits ();
    testDivisibleBy9 ();
    printResult ();
}
```

```
void getAnInteger ()
{
    // Read a positive integer
    do
    {
        cout << "Enter a positive integer: ";
        cin >> n;
    } while (n <= 0);
}
```

```
void getSumOfDigits ()
{
    // Get the sum of digits
```

```

    sumDigits = 0;
    while (n > 0)
    {
        sumDigits = sumDigits + n % 10;
        n = n / 10;
    }
}

void testDivisibleBy9 ()
{
    // Test divisible by 9
    if (sumDigits % 9 == 0)
        ok = true;
    else
        ok = false;
}

void printResult ()
{
    // print out the result
    if (ok)
        cout << "Divisible by 9" << endl;
    else
        cout << "Not divisible by 9" << endl;
}

```

// Passing by values

```
#include <iostream>
using namespace std;

void passingByValues (int);

int main ()
{
    int x = 100;

    passingByValues (x);
    cout << x << endl;    // 100
}

void passingByValues (int x)
{
    x = 200;
}
```

// A positive integer is divisible by 9
// if and only if the sum of digits is
// divisible by 9

```
#include <iostream>
using namespace std;
```

```
int getAnInteger (int);
int getSumOfDigits (int);
bool testDivisibleBy (int, int);
void printResult (bool, int, int);
```

```
int main ()
{
    int n;
    int sumDigits;

    // get an integer greater than 0
    n = getAnInteger (0);

    // get the sum of digits of n
    sumDigits = getSumOfDigits (n);

    // test divisible by 9
    bool ok = testDivisibleBy (sumDigits, 9);

    // print resultt
    printResult (ok, n, 9);
}
```

```
int getAnInteger (int limit)
{
    int n;

    do
    {
        cout << "Enter a positive integer: ";
        cin >> n;
    } while (n <= limit);

    return n;
}
```

```
int getSumOfDigits (int n)
```

```

{
    int sumDigits = 0;

    while (n > 0)
    {
        sumDigits += n % 10;
        n /= 10;
    }

    return sumDigits;
}

bool testDivisibleBy (int sum, int k)
{
    if (sum % k == 0)
        return true;
    else
        return false;
}

void printResult (bool ok, int n, int k)
{
    if (ok)
        cout << n << " is divisible by " << k << endl;
    else
        cout << n << " is not divisible by " << k << endl;
}

```

```

// Properties of integers.
// Given a positive integer n
// 1. Find the sum of odd digits
// n = 12345 ==> 1 + 3 + 5 = 9
// 2. Get the length of the integer
// 3. Get the number of even digits
// 4. Print backward of an integer. For example
// n = 12345
// output: 5    4    3    2    1
// 5. Form the reverse of an integer
// n = 12345 ==> get an integer of 54321
// 6. Print forward an integer. For example
// n = 12345
// output: 1    2    3    4    5
// 7. Any other?
//

```

```

// Read a few sets (may be 2 sets) to test the above
#include <iostream>
using namespace std;

```

```

int getSumOddDigits (int);
int getLength (int);
int getNoEven (int);
void printBackward (int);
int getReverse (int);
void printForward (int);

```

```

// if the digit k is odd, I print * * * ... (k times)
// if the digit k is even, I print & & & .. (k times)
void printPattern (int);

```

```

int main ()
{
    int n;

    for (int i = 1; i <= 2; i++)
    {
        cout << "Enter positive integer " << i << ": ";
        cin >> n;

        cout << "The sum of odd digits in "
             << n
             << " = "
             << getSumOddDigits (n)
    }
}

```

```

        << endl;

    cout << "The length is "
        << getLength (n)
        << endl;

    cout << "The no of even digits is "
        << getNoEven (n)
        << endl;

    printBackward (n);

    cout << "The reverse of " << n << " is "
        << getReverse (n)
        << endl;

    printForward (n);

    printPattern (n);

    cout << "-----" << endl;
}
}

```

```

int getSumOddDigits (int n)
{
    int sumOdd = 0;
    int lastDigit;

    while (n > 0)
    {
        lastDigit = n % 10;

        if (lastDigit % 2 == 1)
            sumOdd += lastDigit;

        n = n / 10;
    }

    return sumOdd;
}

```

```

int getLength (int n)
{

```

```

    int length = 0;

    while (n > 0)
    {
        n /= 10;
        length++;
    }

    return length;
}

int getNoEven (int n)
{
    int countEven = 0;
    int lastDigit;

    while (n > 0)
    {
        lastDigit = n % 10;

        if (lastDigit % 2 == 0)
            ++countEven;

        n /= 10;
    }

    return countEven;
}

void printBackward (int n)
{
    while (n > 0)
    {
        cout << n % 10 << "\t";
        n /= 10;
    }

    cout << endl;
}

```

```

// 1234 = 1 * 1000 + 2 * 100 + 3 * 10 + 4
// 4321 = 4 * 1000 + 3 * 100 + 2 * 10 + 1
int getReverse (int n)

```

```

{
    int reverse = 0;

    while (n > 0)
    {
        reverse = reverse * 10 + n % 10;
        n = n / 10;
    }

    return reverse;
}

void printForward (int n)
{
    //    n = getReverse(n);
    //    printBackward(n);

    printBackward(getReverse(n));
}

void printPattern (int n)
{
    n = getReverse (n);
    int lastDigit;

    while (n > 0)
    {
        lastDigit = n % 10;

        for (int k = 1; k <= lastDigit; k++)
        {
            if (lastDigit % 2)
                cout << "*\t";
            else
                cout << "&\t";
        }

        cout << endl;

        n = n / 10;
    }
}

```

```

// Properties of integers.
// Given a positive integer n
// 1. Find the sum of odd digits
// n = 12345 ==> 1 + 3 + 5 = 9
// 2. Get the length of the integer
// 3. Get the number of even digits
// 4. Print backward of an integer. For example
// n = 12345
// output: 5    4    3    2    1
// 5. Form the reverse of an integer
// n = 12345 ==> get an integer of 54321
// 6. Print forward an integer. For example
// n = 12345
// output: 1    2    3    4    5
// 7. Any other?
//
// Read a few sets (may be 2 sets) to test the above
#include <iostream>
using namespace std;

int getSumOddDigits (int);
int getLength (int);
int getNoEven (int);
void printBackward (int);
int getReverse (int);
void printForward (int);

// if the digit k is odd, I print * * * ... (k times)
// if the digit k is even, I print & & & .. (k times)
void printPattern (int);
void printALine (int, char);

int main ()
{
    int n;

    for (int i = 1; i <= 2; i++)
    {
        cout << "Enter positive integer " << i << ": ";
        cin >> n;

        cout << "The sum of odd digits in "
             << n
             << " = "

```

```

        << getSumOddDigits (n)
    << endl;

    cout << "The length is "
        << getLength (n)
        << endl;

    cout << "The no of even digits is "
        << getNoEven (n)
        << endl;

    printBackward (n);

    cout << "The reverse of " << n << " is "
        << getReverse (n)
        << endl;

    printForward (n);

    printPattern (n);

    cout << "-----" << endl;
}
}

```

```

int getSumOddDigits (int n)
{
    int sumOdd = 0;
    int lastDigit;

    while (n > 0)
    {
        lastDigit = n % 10;

        if (lastDigit % 2 == 1)
            sumOdd += lastDigit;

        n = n / 10;
    }

    return sumOdd;
}

```

```

int getLength (int n)

```

```

{
    int length = 0;

    while (n > 0)
    {
        n /= 10;
        length++;
    }

    return length;
}

int getNoEven (int n)
{
    int countEven = 0;
    int lastDigit;

    while (n > 0)
    {
        lastDigit = n % 10;

        if (lastDigit % 2 == 0)
            ++countEven;

        n /= 10;
    }

    return countEven;
}

void printBackward (int n)
{
    while (n > 0)
    {
        cout << n % 10 << "\t";
        n /= 10;
    }

    cout << endl;
}

```

```

// 1234 = 1 * 1000 + 2 * 100 + 3 * 10 + 4
// 4321 = 4 * 1000 + 3 * 100 + 2 * 10 + 1

```

```

int getReverse (int n)
{
    int reverse = 0;

    while (n > 0)
    {
        reverse = reverse * 10 + n % 10;
        n = n / 10;
    }

    return reverse;
}

```

```

void printForward (int n)
{
    //    n = getReverse(n);
    //    printBackward(n);

    printBackward(getReverse(n));
}

```

```

void printPattern (int n)
{
    n = getReverse (n);
    int lastDigit;

    while (n > 0)
    {
        lastDigit = n % 10;

        if (lastDigit % 2)
            printALine (lastDigit, '*');
        else
            printALine (lastDigit, '&');

        n = n / 10;
    }
}

```

```

void printALine (int n, char ch)
{
    for (int i = 1; i <= n; i++)

```

```
        cout << ch << "\\t";  
    cout << endl;  
}
```

**// parameters pass by value versus
// reference**

```
#include <iostream>
using namespace std;
```

```
void passingByValue (int);
void passingByReference (int&);
```

```
int main ()
{
    int n = 100;

    passingByValue (n);
    cout << "n in main = ";
    cout << n << endl; // 100

    passingByReference (n);
    cout << "n in main = ";
    cout << n << endl; // 200
}
```

```
void passingByValue (int n)
{
    n = n + 100;

    cout << "n in passingByValue function = ";

    cout << n << endl; // 200
}
```

```
// Read in two equal length integers  
// and find the sum of products of their  
// corresponding digits. For example  
// n = 1234, m = 5678  
// sum of products = 1 * 5 + 2 * 6 + 3 * 7 + 4 * 8
```

```
#include <iostream>
```

```
using namespace std;
```

```
void getLength (int, int&);
```

```
void getTwoIntegers (int&, int&);
```

```
void getSumDigitsProduct (int, int, int&);
```

```
int main ()
```

```
{
```

```
    int m, n;
```

```
    getTwoIntegers (m, n);
```

```
    int sumDigitsProduct;
```

```
    getSumDigitsProduct (m, n, sumDigitsProduct);
```

```
    cout << "The sum of digits product = "
```

```
        << sumDigitsProduct
```

```
        << endl;
```

```
}
```

```
void getLength (int n, int& length)
```

```
{
```

```
    length = 1;
```

```
    while (n >= 10)
```

```
    {
```

```
        length++;
```

```
        n /= 10;
```

```
    }
```

```
}
```

```
void getTwoIntegers (int& m, int& n)
```

```
{
```

```
    int lenM, lenN;
```

```
    do
```

```
    {
```

```
        cout << "Enter two integers: ";
```

```

        cin >> m >> n;

        getLength (m, lenM);
        getLength (n, lenN);
    } while (lenM != lenN);
}

void getSumDigitsProduct (int m, int n, int& sumP)
{
    sumP = 0;

    while (m > 0 && n > 0)
    {
        sumP += (m % 10) * (n % 10);

        m /= 10;
        n /= 10;
    }
}

```

**// Given a positive integer, we wish to know
// its length, the sum of odd digits and the
// number of even digits in it.**

```
#include <iostream>
using namespace std;
```

```
int integerInfo (int, int&, int&, int&);
```

```
int main ()
{
    int n;

    cout << "Enter a positive integer: ";
    cin >> n;

    int length, sumOdds, noEvens;

    integerInfo (n, length, sumOdds, noEvens);

    cout << "Length = " << length << endl;
    cout << "Sum odd digits = " << sumOdds << endl;
    cout << "No of even digits = " << noEvens << endl;
}
```

```
int integerInfo (int n, int& length, int& sumOdds, int& noEvens)
{
    length = 0;
    sumOdds = 0;
    noEvens = 0;

    while (n > 0)
    {
        length++;

        if (n % 10 % 2)
            sumOdds += n % 10;
        else
            noEvens++;

        n /= 10;
    }
}
```

// compare static or non-static (auto) variables

```
#include <iostream>
using namespace std;

void testNonStatic ();
void testStatic ();

int main ()
{
    for (int i = 1; i <= 5; i++)
        testNonStatic ();

    cout << "-----" << endl;

    for (int i = 1; i <= 5; i++)
        testStatic ();
}

void testNonStatic ()
{
    int x = 0;

    x += 10;

    cout << x << endl;
}

void testStatic ()
{
    static int x = 0;

    x += 10;

    cout << x << endl;
}
```

// Default parameters

```
#include <iostream>
using namespace std;
```

```
int defaultPara (int x = 10, int y = 20, int z = 30);
```

```
int main ()
{
    int x = 100;
    int y = 200;
    int z = 300;

    cout << defaultPara (x, y, z) << endl; // 600
    cout << defaultPara () << endl;         // 60
    cout << defaultPara (x) << endl;        // 150
    cout << defaultPara (x, y) << endl;     // 330
    cout << defaultPara (z) << endl;       //

}

int defaultPara (int x, int y, int z)
{
    return x + y + z;
}
```

// Default parameters

// Other notations

#include <iostream>

using namespace std;

int defaultPara (int = 10, int = 20, int = 30);

int main ()

{

int x = 100;

int y = 200;

int z = 300;

cout << defaultPara (x, y, z) << endl; // 600

cout << defaultPara () << endl; // 60

cout << defaultPara (x) << endl; // 150

cout << defaultPara (x, y) << endl; // 330

cout << defaultPara (z) << endl; // 350

}

int defaultPara (int x, int y, int z)

{

return x + y + z;

}

// generate some random numbers

```
#include <iostream>
```

```
#include <cstdlib>
```

```
using namespace std;
```

```
int main ()
```

```
{
```

```
    int seed;
```

```
    cout << "Enter a seed: ";
```

```
    cin >> seed;
```

```
    srand (seed);
```

```
    cout << rand () << endl;
```

```
    cout << rand () << endl;
```

```
    cout << rand () << endl;
```

```
    cout << rand () * rand () << endl;
```

```
}
```

// generate some random numbers

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
```

```
int main ()
{
```

```
    srand (time (NULL));
```

```
    cout << rand () << endl;
    cout << rand () << endl;
    cout << rand () << endl;
    cout << rand () * rand () << endl;
```

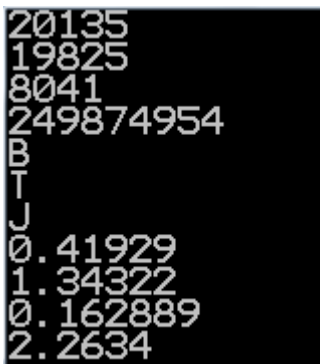
```
    // generate some characters
```

```
    cout << static_cast <char> (rand () % 26 + 65) << endl;
    cout << static_cast <char> (rand () % 26 + 65) << endl;
    cout << static_cast <char> (rand () % 26 + 65) << endl;
```

```
    // generate some real numbers
```

```
    cout << 1.0 * rand () / (rand () + 1) << endl;
    cout << 1.0 * rand () / (rand () + 1) << endl;
    cout << 1.0 * rand () / (rand () + 1) << endl;
    cout << 1.0 * rand () / (rand () + 1) << endl;
```

```
}
```



A screenshot of a terminal window with a black background and white text. The output of the C++ program is displayed line by line, corresponding to the cout statements in the code. The first four lines show random integers: 20135, 19825, 8041, and 249874954. The next three lines show random characters: 'B', 'T', and 'J'. The final four lines show random floating-point numbers: 0.41929, 1.34322, 0.162889, and 2.2634.

```

#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main()
{
    bool more=true;
    int number, input;

    srand(time(NULL));
    number = rand() % 10000;

    cout << "I have a number from 0 to 9999" << endl;
    cout << "Guess what is my number?" << endl;
    cout << endl;

    while (more)
    {
        cout << "Your guess: ";
        cin >> input;

        if (input < number)
            cout << "Number is higher\n";
        else if (input > number)
            cout << "Number is lower\n";
        else
        {
            cout << "Got it!";
            more= false;
        }
    }
    return 0;
}

```

```
I have a number from 0 to 9999
Guess what is my number?

Your guess: 5000
Number is lower
Your guess: 2500
Number is lower
Your guess: 1250
Number is lower
Your guess: 675
Number is lower
Your guess: 325
Number is higher
Your guess: 500
Number is lower
Your guess: 412
Number is lower
Your guess: 380
Number is lower
Your guess: 350
Number is lower
Your guess: 375
Number is lower
Your guess: 335
Number is higher
Your guess: 340
Number is higher
Your guess: 345
Number is lower
Your guess: 342
Number is higher
Your guess: 343
Number is higher
Your guess: 344
Got it!
```

```

// To find the sum from 1 to N
// = To find the sum from N down to 1
//
// Base case:
// sum (1 to 1) = sum (1) = 1
// Genaral case
// sum (1 to N) = sum (N)
//          = N + sum (N - 1)
#include <iostream>
#include <cstdlib>
using namespace std;

```

```
int sum (int);
```

```

int main ()
{
    int n;

    srand (time (NULL));

    for (int i = 1; i <= 5; i++)
    {
        n = rand () % 100 + 1;

        cout << "The sum from 1 to "
              << n << " is "
              << sum (n)
              << endl;
    }
}

```

```

int sum (int n)
{
    if (n == 1)
        return 1;
    else
        return n + sum (n - 1);
}

```

```

// Given positive integer n, we wish to find
// the sum of digits of n
// Base case
// sumDigits (n) = n if n is less than 10
// General case
// sumDigits (n) = n % 10 + sumDigits (n / 10)

```

```

#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

```

```

int sumDigits (int);

```

```

int main ()
{
    int n;

    srand (time (NULL));

    for (int i = 1; i <= 10; i++)
    {
        n = rand () + 1;

        cout << "The sum of digits of "
              << n
              << " is "
              << sumDigits (n)
              << endl;
    }
}

int sumDigits (int n)
{
    if (n < 10)
        return n;
    else
        return n % 10 + sumDigits (n / 10);
}

```

```

// Given positive integer n, we wish to find
// the sum of all the odd digits in n
// Base case
// sumDigits (n) = n if n < 10 and odd
// sumDigits (n) = 0, if n < 10 and even
// General case
// sumDigits (n) = n % 10 + sumDigits (n / 10),
// if n % 10 is odd
// sumDigits (n) = sumDigits (n / 10)
// if n % 10 is even

```

```

#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

```

```

int sumOddDigits (int);

```

```

int main ()
{
    int n;

    srand (time (NULL));

    for (int i = 1; i <= 10; i++)
    {
        n = rand () + 1;

        cout << "The sum of all odd digits in "
              << n
              << " is "
              << sumOddDigits (n)
              << endl;
    }
}

```

```

int sumOddDigits (int n)
{
    if (n < 10 && n % 2 == 1)
        return n;
    else if (n < 10 && n % 2 == 0)
        return 0;
    else if (n % 10 % 2 == 1)
        return n % 10 + sumOddDigits (n / 10);
    else

```

```
        return sumOddDigits (n / 10);  
    }
```

```

// Given a positive integer n, we wish to
// print reverse of that positive integer
// n = 12345 ==> 5*4*3*2*1*
//
// while (n > 0)
// {
//     cout << n % 10 << "*"
//     n /= 10;
// }
//
// cout << endl;
//
// Base case
// printReverse (n)
//     if n = 0, cout << endl;
// General case
// if n > 0
//     cout << n % 10 << "*";
//     printReverse (n / 10)
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

void printReverse (int);

int main ()
{
    int n;

    srand (time (NULL));

    for (int i = 1; i <= 10; i++)
    {
        n = rand ();

        cout << "Print reverse of "
             << n
             << " is ";

        printReverse (n);
    }
}

```

```
void printReverse (int n)
{
    if (n == 0)
        cout << endl;
    else
    {
        cout << n % 10 << "*";
        printReverse (n / 10);
    }
}
```

// shift operators

```
#include <iostream>
using namespace std;

int main ()
{
    int n = 187;
    int m = 47;

    cout << "The size of int is "
         << sizeof (int) * 8
         << " bits"
         << endl;                                // 32

    cout << (n >> 2) << endl;                    // 46
    cout << (m << 2) << endl;                    // 188

    cout << (n & m) << endl;                    // 43
    cout << (n | m) << endl;                    // 191
    cout << (n ^ m) << endl;                    // 148

    cout << ((n >> 2) & (m << 2)) << endl; // 44
}
/*
```

n = 187

```
187 / 2 = 93 R 1
93 / 2 = 46 R 1
46 / 2 = 23 R 0
23 / 2 = 11 R 1
11 / 2 = 5 R 1
5 / 2 = 2 R 1
2 / 2 = 1 R 0
1 / 2 = 0 R 1
```

Binary representtton is
10111011

Internal reprsentation
00000000000000000000000010111011

n >> 2

11

$$\begin{aligned} &= 2^5 + 2^3 + 2^2 + 2^1 \\ &= 32 + 8 + 4 + 2 \\ &= 46 \end{aligned}$$

$$m = 47$$

$47/2 = 23$	R 1
$23/2 = 11$	R 1
$11/2 = 5$	R 1
$5/2 = 2$	R 1
$2/2 = 1$	R 0
$1/2 = 0$	R 1

Binary representation

101111

Internal representation

[illegible]

$$m \ll 2$$

0000000000000000000000000000000010111100

$$\begin{aligned} &= 2^7 + 2^5 + 2^4 + 2^3 + 2^2 \\ &= 128 + 32 + 16 + 8 + 4 \\ &= 160 + 28 \\ &= 188 \end{aligned}$$

$$n \wedge m$$

0000000000000000000000000000000010111011

00000000000000000000000000000000101111 ^

00000000000000000000000000000000000010010100

$$\begin{aligned} &= 2^7 + 2^4 + 2^2 \\ &= 128 + 16 + 4 \\ &= 148 \text{ (ans)} \end{aligned}$$

*/

**// Using shift operators, you can check
// if a positive integer is odd or even**

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main ()
{
    int m, n;

    srand (time (NULL));

    m = rand ();

    n = m;
    n = (n >> 1);           //      n / 2;
    n = (n << 1);           // n * 2;

    if (m == n)
        cout << m << " is even" << endl;
    else
        cout << m << " is odd" << endl;
}
```

```

#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main()
{
    bool more=true;
    int number, input;

    srand(time(NULL));
    number = rand() % 10000;

    cout << "I have a number from 0 to 9999" << endl;
    cout << "Guess what is my number?" << endl;
    cout << endl;

    while (more)
    {
        cout << "Your guess: ";
        cin >> input;

        if (input < number)
            cout << "Number is higher\n";
        else if (input > number)
            cout << "Number is lower\n";
        else
        {
            cout << "Got it!";
            more= false;
        }
    }
    return 0;
}

```

**// An array element can be passed to a function
// by value or by reference**

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

const int MAX = 10;

void getValue (int&);
void printValue (int);

int main ()
{
    int a [MAX];

    srand (time (NULL));

    for (int i = 0; i < MAX; i++)
        getValue (a [i]);

    for (int i = 0; i < MAX; i++)
        printValue (a [i]);
    cout << endl;
}

void getValue (int& n)
{
    n = rand () % 100;
}

void printValue (int n)
{
    cout << n << "\t";
}
```

// Construct an array of characters

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

const int MAX = 10;

int main ()
{
    char a [MAX];

    srand (time (NULL));

    for (int i = 0; i < MAX; i++)
        a [i] = static_cast <char>(rand () % 26 + 65);

    for (int i = 0; i < MAX; i++)
        cout << a [i] << "\t";
    cout << endl;
}
```

// Passing an array to a function.
// Array is passed by reference

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

const int MAX = 10;

void constructArray (int [], int);
void printArray (int [], int);

int main ()
{
    int a [MAX];

    srand (time (NULL));

    constructArray (a, MAX);
    printArray (a, MAX);
}

void constructArray (int a [], int size)
{
    for (int i = 0; i < size; i++)
        a [i] = rand () % 100;
}

void printArray (int a [], int size)
{
    for (int i = 0; i < size; i++)
        cout << a [i] << "\t";
    cout << endl;
}
```

// Passing an array to a function.
// Array is passed by reference
//
// construct more arrays of different sizes

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

const int MAX = 10;

void constructArray (int [], int);
void printArray (int [], int);

int main ()
{
    int a [MAX];

    srand (time (NULL));

    for (int i = 1; i <= 5; i++)
    {
        int size = rand () % 10 + 1;
        constructArray (a, size);
        printArray (a, size);
    }
}

void constructArray (int a [], int size)
{
    for (int i = 0; i < size; i++)
        a [i] = rand () % 100;
}

void printArray (int a [], int size)
{
    for (int i = 0; i < size; i++)
        cout << a [i] << "\t";
    cout << endl;
}
```

79	82	77	18	32	
87	32				
35	13				
46	30	82	13	11	
46	65	43	63	43	14

// Passing an array to a function.
// Array is passed by reference
//
// For each of the array generated, we wish
// to know the smallest and the largest value

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

const int MAX = 10;

void constructArray (int [], int);
void printArray (const int [], int);

void findMinMax (const int [], int, int&, int&);

int main ()
{
    int a [MAX];

    srand (time (NULL));

    int min, max;

    for (int i = 1; i <= 5; i++)
    {
        int size = rand () % 10 + 1;
        constructArray (a, size);
        printArray (a, size);

        findMinMax (a, size, min, max);
        cout << "Minimum = " << min << endl;
        cout << "Maximum = " << max << endl;

        cout << "-----" << endl;
    }
}

void constructArray (int a [], int size)
{
    for (int i = 0; i < size; i++)
```

```

        a [i] = rand () % 100;
    }

void printArray (const int a [], int size)
{
    // a [5] = 123;
    // attempt to change the change the value
    // compile time error
    for (int i = 0; i < size; i++)
        cout << a [i] << "\t";
    cout << endl;
}

void findMinMax (const int a [], int size, int& min, int& max)
{
    min = a [0];
    max = a [0];

    for (int i = 1; i < size; i++)
    {
        if (min > a [i])
            min = a [i];

        if (max < a [i])
            max = a [i];
    }
}

```

```

14
Minimum = 14
Maximum = 14
-----
37      97      7      28      18
Minimum = 7
Maximum = 97
-----
76      35      27      76      70      10      42      24
Minimum = 10
Maximum = 76
-----
96      51      53      64      78      96      61
Minimum = 51
Maximum = 96
-----
41      8      67      6      61
Minimum = 6
Maximum = 67
-----

```

What is the effect of this program segment?

```
for (int i = 0; i < MAX_SIZE/2; ++i)
{
    int temp = a[i];
    a[i] = a [MAX_SIZE - i - 1];
    a[MAX_SIZE - i - 1 ] = temp;
}
```

```
int a [6] = { 1, 2, 3, 4, 5, 6};
MAX_SIZE = 6
MAX_SIZE / 2 = 3
```

```
i      i < 3      ....
0      0 < 3? yes  temp = a [0];
                   a [0] = a [5];
                   a [5] = temp;
```

a ==> {6, 2, 3, 4, 5, 1}

```
1      1 < 3? yes  temp = a [1];
                   a [1] = a [4];
                   a [4] = temp;
```

a ==> {6, 5, 3, 4, 2, 1}

```
2
a ==> (6, 5, 4, 3, 2, 1}
```

What is the effect of this program segment?

```
for(int i = 0; i < MAX_SIZE -1; ++i)
{
    if (a[i] > a[i + 1])
    {
        temp = a[i];
        a[i] = a[i + 1];
        a[i + 1] = temp;
    }
}
```

```

}

int a [] = {44, 22, 32, 55, 27};

MAXSIZE = 5;

i      a [i] > a [i - 1]?    ....

0      a [0] > a [1]? y      temp = a [0];
                                a [0] = a [1];
                                a [1] = temp;

                                a ==> 22, 44, 32, 55, 27

1      a [1] > a[2]? y          temp = a [1];
                                a [1] = a [2];
                                a [2] = temp;

                                a ==> 22, 32, 44, 55, 27

2      a [2] > a [3]? N

3      a [3] > a [4]? y
                                a ==> 22, 32, 44, 27, 55
-----

```

// A two D array

// - construct this array row by row

```
#include <iostream>
```

```
#include <ctime>
```

```
#include <cstdlib>
```

```
using namespace std;
```

```
const int MAXROW = 10;
```

```
const int MAXCOL = 8;
```

```
void constructARow (int [], int);
```

```
void printARow (const int [], int);
```

```
int main ()
```

```
{
```

```
    int a [MAXROW][MAXCOL];
```

```
    srand (time (NULL));
```

```
    int row = rand () % 5 + 6;
```

```
    int column = rand () % 4 + 5;
```

```
    for (int i = 0; i < row; i++)
```

```
        constructARow (a [i], column);
```

```
    for (int i = 0; i < row; i++)
```

```
        printARow (a [i], column);
```

```
}
```

```
void constructARow (int x [], int size)
```

```
{
```

```
    for (int i = 0; i < size; i++)
```

```
        x [i] = rand () % 100;
```

```
}
```

```
void printARow (const int x [], int size)
```

```
{
```

```
    for (int i = 0; i < size; i++)
```

```
        cout << x [i] << "\t";
```

```
    cout << endl;
```

```
}
```

25	93	39	53	51	47
93	49	30	99	1	93
47	8	27	4	44	83
32	27	63	29	33	70
39	11	73	80	57	27
51	38	4	63	7	23

```

// A two D array
// - construct a 2D array
// - print a 2D array
// - look for the largest and the smallest elements
#include <iostream>
#include <ctime>
#include <cstdlib>
using namespace std;

const int MAXROW = 10;
const int MAXCOL = 8;

void construct2D (int [][][MAXCOL], int, int);
void print2D (const int [][][MAXCOL], int, int);
void findMinMax (const int [][][MAXCOL], int, int, int&, int&);

int main ()
{
    int a [MAXROW][MAXCOL];

    srand (time (NULL));

    for (int i = 1; i <= 3; i++)
    {
        int row = rand () % 5 + 6;
        int column = rand () % 4 + 5;

        construct2D (a, row, column);
        print2D (a, row, column);

        int min, max;
        findMinMax (a, row, column, min, max);
        cout << endl;
        cout << "minimum = " << min << endl;
        cout << "maximum = " << max << endl;

        cout << "-----" << endl;
    }
}

void construct2D (int a [][][MAXCOL], int row, int column)
{
    for (int i = 0; i < row; i++)

```

```

    {
        for (int j = 0; j < column; j++)
        {
            a [i][j] = rand () % 1000;
        }
    }
}

```

```

void print2D (const int a[][MAXCOL], int row, int column)
{
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < column; j++)
        {
            cout << a [i][j] << "\t";
        }

        cout << endl;
    }
}

```

```

void findMinMax (const int a[][MAXCOL], int row, int column,
                 int& min, int& max)
{
    min = a [0][0];
    max = a [0][0];

    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < column; j++)
        {
            if (min > a [i][j])
                min = a [i][j];

            if (max < a [i][j])
                max = a [i][j];
        }
    }
}

```

```

312      808      973      273      365
455      214      961      768      906
406      538      570      842      43
785      800      615      759      792
802      640      189      654      879
962      349      606      111      796

minimum = 43
maximum = 973
-----
41        56        415        178        463        299        712        112
766       89        730        814        730        427        458        626
706       514       767        136        104        81        508        824
292       630       107        682        113        457        372        816
351       248       30         146        763        882        645        856
444       10        813        331        303        671        510        157

minimum = 10
maximum = 882
-----
816       187       236       319       409       783       820
440       669       883       274       86        349       579
58        593       83        738       865       218       621
488       265       7         232       322       16        815
294       746       618       335       915       737       942
548       741       18        191       401       738       521
244       787       479       156       346       803       398
513       657       444       423       920       220       687

minimum = 7
maximum = 942
-----

```

```

// A two D array
// - construct a 2D array of some ASCII characters
// - print a 2D array
// - look for the largest and the smallest elements
#include <iostream>
#include <ctime>
#include <cstdlib>
using namespace std;

const int MAXROW = 10;
const int MAXCOL = 8;

void construct2D (char [][][MAXCOL], int, int);
void print2D (const char [][][MAXCOL], int, int);
void findMinMax (const char [][][MAXCOL], int, int, char&, char&);

int main ()
{
    char a [MAXROW][MAXCOL];

    srand (time (NULL));

    for (int i = 1; i <= 3; i++)
    {
        int row = rand () % 5 + 6;
        int column = rand () % 4 + 5;

        construct2D (a, row, column);
        print2D (a, row, column);

        char min, max;
        findMinMax (a, row, column, min, max);
        cout << endl;
        cout << "minimum = " << "\"" << min << "\"" << endl;
        cout << "maximum = " << "\"" << max << "\"" << endl;

        cout << "-----" << endl;
    }
}

void construct2D (char a [][][MAXCOL], int row, int column)
{
    for (int i = 0; i < row; i++)

```

```

{
    for (int j = 0; j < column; j++)
    {
        a [i][j] = static_cast <char>(rand () % 128);
    }
}

```

```

void print2D (const char a[][MAXCOL], int row, int column)
{
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < column; j++)
        {
            cout << "\" << a [i][j] << "\" << "\"t";
        }

        cout << endl;
    }
}

```

```

void findMinMax (const char a[][MAXCOL], int row, int column,
                 char& min, char& max)
{
    min = a [0][0];
    max = a [0][0];

    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < column; j++)
        {
            if (min > a [i][j])
                min = a [i][j];

            if (max < a [i][j])
                max = a [i][j];
        }
    }
}

```

CSCI114 – Procedural Programming
Class Examples – Last update: 21 February 2017
107

```

// An (entire) array is only passed by reference
//
// construct a few array of random sizes between 5 to 10
//
// note the use of "const" in parameter
//
// A recursive function to count and to return the number of odd digits

```

```

#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

const int MAX = 10;

// array + size of the array (the 2 parameters)
void constructArray (int [], int);

void printArray (const int [], int);

int countOdd (const int [], int);

int main ()
{
    int a [MAX];

    int size;

    srand (time (NULL));

    int largest, smallest;

    for (int i = 1; i <= 5; i++)
    {
        size = rand () % 2 + 9;

        constructArray (a, size);
        printArray (a, size);

        cout << "The array has " << countOdd (a, size)
              << " odd integers"
              << endl;
    }
}

```

```

        cout << "-----" << endl;
    }
}

```

```

void constructArray (int a [], int size)
{
    for (int i = 0; i < size; i++)
        a [i] = rand () % 100;
}

```

```

void printArray (const int a [], int size)
{
    for (int i = 0; i < size; i++)
        cout << a [i] << "\t";
    cout << endl;
}

```

```

int countOdd (const int a [], int size)
{
    if (size == 0)
        return 0;
    else if (a [size - 1] % 2 == 1)
        return 1 + countOdd (a, size - 1);
    else
        return countOdd (a, size - 1);
}

```

```

// A C-string - character array
//
// Given a string
// - how many uppercase letters?
// - how many lowercase letters?
// - how many digits?
// - how many spaces?
// - extract the uppercase and the lowercase letters and
//   return two substrings
// - change all uppercase to lowercase letters and vice versa
//   all others, unchange
// - find sum digits stored in the cstring

```

```

#include <iostream>
#include <cctype>
using namespace std;

```

```

const int MAX = 80;

```

```

void stringInfo (const char [], int&, int&, int&, int&);
void getUpperLowerStr (const char [], char [], char []);
void updateStr (char []);
int getSumDigits (char []);

```

```

int main ()

```

```

{
    char str [MAX];

```

```

    cout << "Enter a string: ";
    cin.getline (str, MAX);
    cout << str << endl;

```

```

    int noUppers,
        noLowers,
        noDigits,
        noSpaces;

```

```

    stringInfo (str, noUppers, noLowers, noDigits, noSpaces);

```

```

    cout << "No of uppers = " << noUppers << endl;
    cout << "No of lowers = " << noLowers << endl;
    cout << "No of digits = " << noDigits << endl;

```

```

cout << "No of spaces = " << noSpaces << endl;

char lowerStr [MAX];
char upperStr [MAX];

getUpperLowerStr (str, upperStr, lowerStr);

cout << upperStr << endl;
cout << lowerStr << endl;

updateStr (str);
cout << str << endl;

cout << "The sum of digits in the string = "
      << getSumDigits (str)
      << endl;
}

void stringInfo (const char str [], int& noUppers,
                int& noLowerers, int& noDigits,
                int& noSpaces)
{

    noUppers = 0;
    noLowerers = 0;
    noDigits = 0;
    noSpaces = 0;

    int i = 0;

    while (str [i] != '\0')
    {
        if (isupper (str [i]))
            ++noUppers;
        else if (islower (str [i]))
            ++noLowerers;
        else if (isdigit (str [i]))
            ++noDigits;
        else if (isspace (str [i]))
            ++noSpaces;
    }
}

```

```

        ++i;
    }

}

void getUpperLowerStr (const char str [], char upperStr[], char lowerStr[])
{
    int i = 0;
    int iU = 0;
    int iL = 0;

    while (str [i] != '\0')
    {
        if (isupper (str [i]))
        {
            upperStr [iU] = str [i];
            ++iU;
        }
        else if (islower (str [i]))
        {
            lowerStr [iL] = str [i];
            ++iL;
        }

        ++i;
    }

    upperStr [iU] = '\0';
    lowerStr [iL] = '\0';
}

void updateStr (char str [])
{
    int i = 0;

    while (str [i] != '\0')
    {
        if (isupper (str [i]))
        {
            str [i] = tolower (str [i]);
        }
        else if (islower (str [i]))

```

```

        {
            str [i] = toupper (str [i]);
        }

        ++i;
    }
}

int getSumDigits (char str [])
{
    int i = 0;
    int sumDigits;

    while (str [i] != '\0')
    {
        if (isdigit (str [i]))
            sumDigits = sumDigits + str [i] - '0';

        ++i;
    }

    return sumDigits;
}

```

```

// An integer is symmetric if and only if you
// read the integer from left to right and
// from right to left, they are the same
//
// 12321    123321    8 symmetric
// 123421    no symmetric
//
// Using two arrays, we can check if an integer is symmetric

```

```

#include <iostream>
using namespace std;

```

```

const int MAX = 10;

```

```

bool isSymmetric (int);
int length (int);
void construct2Arrays (int [], int [], int, int);
bool compareArrays (const int [], const int [], int, int);

```

```

int main ()
{
    int n;

    cout << "Enter an integer: ";
    cin >> n;

    if (isSymmetric (n))
        cout << n << " is symmetric" << endl;
    else
        cout << n << " is not symmeitric" << endl;
}

```

```

bool isSymmetric (int n)
{
    int len = length (n);

    int a [MAX];
    int b [MAX];

    construct2Arrays (a, b, n, len);

    bool ok = compareArrays (a, b, len, len);

    if (ok)

```

```

        return true;
    else
        return false;
}

int length (int n)
{
    if (n < 10)
        return 1;
    else
        return 1 + length (n / 10);
}

void construct2Arrays (int a [], int b [], int n, int len)
{
    int i = 0;

    while (n > 0)
    {
        a [i] = n % 10;
        b [len - i - 1] = n % 10;
        ++i;
        n /= 10;
    }
}

bool compareArrays (const int a [], const int b [], int sizeA, int sizeB)
{
    if (sizeA != sizeB)
        return false;

    for (int i = 0; i < sizeA; i++)
        if (a [i] != b [i])
            return false;

    return true;
}

```

**// An integer is symmetric if and only if you
// read the integer from left to right and
// from right to left, they are the same
//
// 12321 123321 8 symmetric
// 123421 no symmetric**

```
//  
// Using two arrays, we can check if an integer is symmetric  
// Recursive version
```

```
#include <iostream>  
using namespace std;
```

```
const int MAX = 10;
```

```
bool isSymmetric (int);  
int length (int);  
void construct2Arrays (int [], int [], int, int);  
bool compareArrays (const int [], const int [], int, int);
```

```
int main ()  
{  
    int n;  
  
    cout << "Enter an integer: ";  
    cin >> n;  
  
    if (isSymmetric (n))  
        cout << n << " is symmetric" << endl;  
    else  
        cout << n << " is not symmeitric" << endl;  
}
```

```
bool isSymmetric (int n)  
{  
    int len = length (n);  
  
    int a [MAX];  
    int b [MAX];  
  
    construct2Arrays (a, b, n, len);  
  
    bool ok = compareArrays (a, b, len, len);  
  
    if (ok)  
        return true;  
    else  
        return false;  
}
```

```

int length (int n)
{
    if (n < 10)
        return 1;
    else
        return 1 + length (n / 10);
}

void construct2Arrays (int a [], int b [], int n, int len)
{
    int i = 0;

    while (n > 0)
    {
        a [i] = n % 10;
        b [len - i - 1] = n % 10;
        ++i;
        n /= 10;
    }
}

// compare recursively

bool compareArrays (const int a [], const int b [],
                    int sizeA, int sizeB)
{
    if (sizeA != sizeB)
        return false;
    else if (sizeA == 0 && sizeB == 0)
        return true;
    else if (a [sizeA - 1] != b [sizeB - 1])
        return false;
    else
        return compareArrays (a, b, --sizeA, --sizeB);
}

```

// Enumeration constants for UOW grades

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

enum UOWGrade {HDist, Dist, Credit, Pass, Fail};

UOWGrade getAGrade (int);
void printAGrade (UOWGrade);

int main ()
{
    int exam;

    srand (time (NULL));

    for (int i = 1; i <= 10; i++)
    {
        exam = rand () % 101;

        cout << "Exam = "
              << exam << " ==> "
              << "Grade = ";

        printAGrade (getAGrade (exam));
    }
}

UOWGrade getAGrade (int exam)
{
    if (exam >= 85)
        return HDist;
    else if (exam >= 75)
        return Dist;
    else if (exam >= 65)
        return Credit;
    else if (exam >= 50)
        return Pass;
    else
        return Fail;
}
```

```
void printAGrade (UOWGrade g)
{
    switch (g)
    {
        case HDist: cout << "HDist" << endl;
                    break;
        case Dist:  cout << "Dist" << endl;
                    break;
        case Credit: cout << "Credit" << endl;
                    break;
        case Pass : cout << "Pass" << endl;
                    break;
        default   : cout << "Fail" << endl;
    }
}
```

// A rectangle structure

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

struct Rectangle
{
    int length;
    int width;
    int area;
    int perimeter;
};

int main ()
{
    Rectangle r;

    srand (time (NULL));

    for (int i = 1; i <= 5; i++)
    {
        r.length = rand () % 1000 + 1;
        r.width = rand () % 1000 + 1;

        r.area = r.length * r.width;
        r.perimeter = 2 * (r.length + r.width);

        cout << "R ("
             << r.length << ", "
             << r.width
             << ")"
             << endl;

        cout << "Area = " << r.area << endl;
        cout << "Perimeter = " << r.perimeter << endl;
        cout << "-----" << endl;
    }
}
```

// An array of rectangles

```
#include <iostream>
#include <ctime>
#include <cstdlib>
using namespace std;

const int MAX = 20;

struct Rectangle
{
    int length;
    int width;
};

int main ()
{
    Rectangle r [MAX];

    srand (time (NULL));

    int size = rand () % 11 + 10;

    for (int i = 0; i < size; i++)
    {
        r [i].length = rand () + 1;
        r [i].width = rand () + 1;
    }

    for (int i = 0; i < size; i++)
    {
        cout << "R ("
            << r [i].length << ", "
            << r [i].width
            << ")"
            << endl;
    }
}
```

/*

**A struct variable can be passed as a parameter
either by value or by reference**

A function can return a value of the type struct

***/**

```
#include <iostream>
#include <ctime>
#include <cstdlib>
using namespace std;

const int MAX = 20;

struct Rectangle
{
    int length;
    int width;
    int area;
    int perimeter;
};

Rectangle getARectangle ();
void updateARectangle (Rectangle&);
void printARectangle (Rectangle);

int main ()
{
    Rectangle r [MAX];

    srand (time (NULL));

    int size = rand () % 11 + 10;

    for (int i = 0; i < size; i++)
        r [i] = getARectangle ();

    for (int i = 0; i < size; i++)
        updateARectangle (r [i]);

    for (int i = 0; i < size; i++)
        printARectangle (r [i]);
}

Rectangle getARectangle ()
{
```

```

    Rectangle r;

    r.length = rand () % 1000 + 1;
    r.width  = rand () % 1000 + 1;

    return r;
}

void updateARectangle (Rectangle& r)
{
    r.area = r.length * r.width;
    r.perimeter = 2 * (r.length + r.width);
}

void printARectangle (Rectangle r)
{
    cout << "R ("
        << r.length << ", " << r.width << ", "
        << r.area << ", " << r.perimeter << ")"
        << endl;
}

```

```
// An array of Rational number structure
//
// A struct variable can be passed as a parameter
// either by value or by reference
//
// A function can return a value of the type struct
//
// Most Important Notes: How to access to nested structures?
```

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
```

```
const int MAX = 30;
```

```
struct Info
{
    int quotient;
    int remainder;
    float value;
    char label [MAX];
};
```

```
struct RationalNo
{
    int numerator;
    int denominator;
    Info ri;
};
```

```
// return type is a structure
RationalNo getARationalNo ();
```

```
// pass by reference
void updateARationalNo (RationalNo&);
```

```
// pass by value
void printARationalNo (RationalNo);
```

```
int main ()
{
    RationalNo r [MAX];
```

```

srand (time (NULL));

int size = rand () % 15 + 16;

for (int i = 0; i < size; i++)
{
    r [i] = getARationalNo ();
    updateARationalNo (r [i]);
    printARationalNo (r [i]);
}
}

// return type is a structure
RationalNo getARationalNo ()
{
    RationalNo r;

    r.numerator = rand ();
    r.denominator = rand () + 1;

    return r;
}

// pass by reference
void updateARationalNo (RationalNo& r)
{
    r.ri.quotient = r.numerator / r.denominator;
    r.ri.remainder = r.numerator % r.denominator;
    r.ri.value = 1.0 * r.numerator / r.denominator;
    strcpy (r.ri.label , "Rational No");
}

// pass by value
void printARationalNo (RationalNo r)
{
    static int i = 0;

    cout << r.ri.label << " " << ++i << " ("
        << r.numerator << ", "
        << r.denominator << ", "
        << r.ri.quotient << ", "
        << r.ri.remainder << ", "
        << r.ri.value << ")"

```

```
}          << endl;
```

```
// Read in the info from a text file
// called "infile.txt" and display
// the sqrt of all values stored in "infile.txt"
// and display the results on the monitor screen
```

```
#include <iostream>
#include <fstream>
#include <cmath>
using namespace std;
```

```
int main ()
{
    ifstream infile;
    infile.open ("infile.txt");

    int x;

    while (infile >> x)
    {
        cout << "The sqrt of " << x
            << " is "
            << sqrt (x)
            << endl;
    }
}
```

```
// Read in the info from a text file  
// called "infile.txt" and display  
// the sqrt of all values stored in "infile.txt"  
// and store the results permanently in an  
// outfile called "outfile.txt"
```

```
//
```

```
// Note the closing of files
```

```
#include <fstream>
```

```
#include <cmath>
```

```
using namespace std;
```

```
int main ()
```

```
{
```

```
    ifstream infile;
```

```
    infile.open ("infile.txt");
```

```
    ofstream outfile;
```

```
    outfile.open ("outfile.txt");
```

```
    int x;
```

```
    while (infile >> x)
```

```
    {
```

```
        outfile << "The sqrt of " << x
```

```
        << " is "
```

```
        << sqrt (x)
```

```
        << endl;
```

```
    }
```

```
    infile.close ();
```

```
    outfile.close ();
```

```
}
```

```
// Read in the info from a text file  
// called "infile.txt" and display  
// the sqrt of all values stored in "infile.txt"  
// and store the results permanently in an  
// outfile called "outfile.txt"  
//  
// Note the closing of files  
//  
// Note the test on file opening
```

```
#include <iostream>  
#include <fstream>  
#include <cmath>  
using namespace std;  
  
int main ()  
{  
    ifstream infile;  
    infile.open ("infile.txt");  
  
    if (!infile)  
    {  
        cout << "File opened for reading failed" << endl;  
        return 1;  
    }  
  
    cout << "File opened for reading OK" << endl;  
  
    ofstream outfile;  
    outfile.open ("outfile.txt");  
  
    if (!outfile)  
    {  
        cout << "Outfile opened for creation failed" << endl;  
        return 1;  
    }  
  
    cout << "Begin the creation of output file" << endl;  
  
    int x;  
  
    while (infile >> x)  
    {
```

```
        outfile << "The sqrt of " << x
            << " is "
                << sqrt (x)
                << endl;
    }

    infile.close ();
    cout << "File closed for reading" << endl;

    outfile.close ();
    cout << "Output file generated" << endl;

    return 0;
}
```

```

// Read in the info from a text file
// called "infile.txt" and display
// the sqrt of all values stored in "infile.txt"
// and store the results permanently in an
// outfile called "outfile.txt"
//
// Note the closing of files
//
// Note the test on file opening

#include <iostream>
#include <fstream>
#include <cmath>
using namespace std;

int main ()
{
    ifstream infile;
    infile.open ("infile.txt");

    if (!infile)
    {
        cout << "File opened for reading failed" << endl;
        return 1;
    }

    cout << "File opened for reading OK" << endl;

    ofstream outfile;
    outfile.open ("outfile.txt");

    if (!outfile)
    {
        cout << "Outfile opened for creation failed" << endl;
        return 1;
    }

    cout << "Begin the creation of output file" << endl;

    int x;

    // If this is not the eof of the file
    // ....

```

**// Using this syntax to read the data
// file, it may lead to infinite loop
// if invalid data occurs in the file!!!**

```
while (!infile.eof ())
{
    infile >> x;

    outfile << "The sqrt of " << x
        << " is "
        << sqrt (x)
        << endl;
}

infile.close ();
cout << "File closed for reading" << endl;

outfile.close ();
cout << "Output file generated" << endl;

return 0;
}
```

```

// Read in the info from a text file
// called "infile.txt" and display
// the sqrt of all values stored in "infile.txt"
// and store the results permanently in an
// outfile called "outfile.txt"
//
// Note the closing of files
//
// Note the test on file opening
//
// Note formatted output
#include <iostream>
#include <fstream>
#include <iomanip>
#include <cmath>
using namespace std;

const int MAX = 60;

int main ()
{
    char fileName [MAX];

    ifstream infile;
    infile.open ("infile.txt");

    if (!infile)
    {
        cout << "File opened for reading failed" << endl;
        return 1;
    }

    cout << "File opened for reading OK" << endl;

    cout << "Enter output file name: ";
    cin >> fileName;

    ofstream outfile;
    outfile.open (fileName);

    if (!outfile)
    {

```

```

        cout << fileName << " opened for creation failed" << endl;
        return 1;
    }

    cout << "Begin the creation of output file  called "
        << fileName
        << endl;

    int x;

    outfile << fixed << showpoint << setprecision (5);

    // This way of reading is ok when
    // invalid data occurs in the input
    // file
    while (infile >> x)
    {
        outfile << "The sqrt of " << x
            << " is "
            << sqrt (x)
            << endl;
    }

    if (infile.eof ())
        cout << "All data processed" << endl;
    else
        cout << "Not all data processed, "
            << " errors may occur in data" << endl;

    infile.close ();
    cout << "File closed for reading" << endl;

    outfile.close ();
    cout << "Output file generated" << endl;

    return 0;
}

```

```

Enter rational no file name: abc.txt
abc.txt opened for creation OK
abc.txt successfully created
-----
Begin the transfer from abc.txt to array
abc.txt closed for the transfer to array
-----
Begin the processing of array
Array was processed
-----
Enter the output file name: xyz.txt
Begin array to outfile
xyz.txt generated

```

abc - Notepad

File	Edit	Format	View	Help
26572	22079	Rational	No	1
2112	29671	Rational	No	2
6858	28776	Rational	No	3
24160	15301	Rational	No	4
20434	29718	Rational	No	5
27042	26041	Rational	No	6
27604	2622	Rational	No	7
13066	25477	Rational	No	8
8613	15583	Rational	No	9
29376	27191	Rational	No	10

xyz - Notepad

File	Edit	Format	View	Help
26572	22079	1	4493	1.2035
2112	29671	0	2112	0.0712
6858	28776	0	6858	0.2383
24160	15301	1	8859	1.5790
20434	29718	0	20434	0.6876
27042	26041	1	1001	1.0384
27604	2622	10	1384	10.5278
13066	25477	0	13066	0.5129
8613	15583	0	8613	0.5527
29376	27191	1	2185	1.0804

// File processing + array of structures

// Same as previous example, with one cstring inside the structure

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <ctime>
#include <iomanip>
using namespace std;
```

```
const int MAX = 100;
```

```
struct RationalNo
{
    int numerator;
    int denominator;
    int quotient;
    int remainder;
    float value;
    char label [MAX];
};
```

```
// 1. Create a text file that describes
```

```
// some rational numbers
```

```
void constructInfile (fstream&, char []);
```

```
// 2. From text file to array of rational numbers
```

```
int fileToArray (fstream&, char [], RationalNo []);
```

```
// 3. Process the array
```

```
void processArray (RationalNo [], int);
```

```
// 4. Array to output file
```

```
void arrayToOutfile (RationalNo [], int, ofstream&, char []);
```

```
int main ()
```

```
{
```

```
    fstream afile;
```

```
    char fileName [MAX];
```

```
    cout << "Enter rational no file name: ";
```

```
    cin >> fileName;
```

```
    srand (time (NULL));
```

```

    constructInfile (afile, fileName);

    cout << "-----" << endl;

    RationalNo r [MAX];

    int size = fileToArray (afile, fileName, r);

    cout << "-----" << endl;

    processArray (r, size);

    cout << "-----" << endl;

    ofstream outfile;

    cout << "Enter the output file name: ";
    cin>> fileName;

    arrayToOutfile (r, size, outfile, fileName);
}

void constructInfile (fstream& afile, char fileName [])
{
    afile.open (fileName, ios::out);

    if (!afile)
    {
        cout << fileName
             << " opened for creation failed"
             << endl;

        exit (-1);
    }

    cout << fileName << " opened for creation OK" << endl;

    int size = rand () % 51 + 50;

    for (int i = 1; i <= size; i++)
    {

```

```

        afile << rand () + 1 << "\t"
            << rand () + 1 << "\t"
            << "Rational No "
            << i
            << endl;
    }

    afile.close ();

    cout << fileName << " successfully created" << endl;
}

int fileToArray (fstream& afile, char fileName [], RationalNo r [])
{
    afile.open (fileName, ios::in);

    if (!afile)
    {
        cout << fileName
            << " opened for reading failed"
            << endl;

        exit (-1);
    }

    cout << "Begin the transfer from " << fileName
        << " to array"
        << endl;

    int i = 0;

    char tabkey;

    while (afile >> r [i].numerator >> r [i].denominator)
    {
        afile.get (tabkey); // read and throw away

        afile.getline (r [i].label, MAX);

        ++i;
    }

    afile.close ();
}

```

```

    cout << fileName << " closed for the transfer to array" << endl;

    return i;
}

void processArray (RationalNo r [], int size)
{
    cout << "Begin the processing of array" << endl;

    for (int i = 0; i < size; i++)
    {
        r [i].quotient = r [i].numerator / r [i].denominator;
        r [i].remainder = r [i].numerator % r [i].denominator;
        r [i].value = 1.0 * r [i].numerator / r [i].denominator;
    }

    cout << "Array was processed" << endl;
}

void arrayToOutfile (RationalNo r [], int size,
                    ofstream& outfile, char fileName [])
{
    outfile.open (fileName);

    if (!outfile)
    {
        cout << fileName
             << " opened for array transfer failed"
             << endl;

        exit (-1);
    }

    cout << "Begin array to outfile" << endl;

    outfile << fixed << showpoint << setprecision (4);

    for (int i = 0; i < size; i++)
    {
        outfile << r [i].label << ":"
                << r [i].numerator << "\t"
                << r [i].denominator << "\t"
                << r [i].quotient << "\t"

```

```
        << r [i].remainder << "\t"
        << r [i].value
        << endl;
    }

    outfile.close ();
    cout << fileName << " generated" << endl;
}
```

// File processing + array of structures

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <ctime>
#include <iomanip>
using namespace std;
```

```
const int MAX = 100;
```

struct RationalNo

```
{
    int numerator;
    int denominator;
    int quotient;
    int remainder;
    float value;
};
```

// 1. Create a text file that describes

// some rational numbers

```
void constructInfile (fstream&, char []);
```

// 2. From text file to array of rational numbers

```
int fileToArray (fstream&, char [], RationalNo []);
```

// 3. Process the array

```
void processArray (RationalNo [], int);
```

// 4. Array to output file

```
void arrayToOutfile (RationalNo [], int, ofstream&, char []);
```

```
int main ()
```

```
{
    fstream afile;
    char fileName [MAX];

    cout << "Enter rational no file name: ";
    cin >> fileName;

    srand (time (NULL));

    constructInfile (afile, fileName);
```

```

    cout << "-----" << endl;

    RationalNo r [MAX];

    int size = fileToArray (afile, fileName, r);

    cout << "-----" << endl;

    processArray (r, size);

    cout << "-----" << endl;

    ofstream outfile;

    cout << "Enter the output file name: ";
    cin>> fileName;

    arrayToOutfile (r, size, outfile, fileName);
}

void constructInfile (fstream& afile, char fileName [])
{
    afile.open (fileName, ios::out);

    if (!afile)
    {
        cout << fileName
             << " opened for creation failed"
             << endl;

        exit (-1);
    }

    cout << fileName << " opened for creation OK" << endl;

    int size = rand () % 51 + 50;

    for (int i = 1; i <= size; i++)
    {
        afile << rand () + 1 << "\t"
             << rand () + 1 << "\t"

```

```

        << "Rational No "
        << i
        << endl;
    }

    afile.close ();

    cout << fileName << " successfully created" << endl;
}

int fileToArray (fstream& afile, char fileName [], RationalNo r [])
{
    afile.open (fileName, ios::in);

    if (!afile)
    {
        cout << fileName
            << " opened for reading failed"
            << endl;

        exit (-1);
    }

    cout << "Begin the transfer from " << fileName
        << " to array"
        << endl;

    int i = 0;

    while (afile >> r [i].numerator >> r [i].denominator)
    {
        afile.clear ();
        afile.ignore (100, '\n');

        ++i;
    }

    afile.close ();
    cout << fileName << " closed for the transfer to array" << endl;

    return i;
}

```

```

void processArray (RationalNo r [], int size)
{
    cout << "Begin the processing of array" << endl;

    for (int i = 0; i < size; i++)
    {
        r [i].quotient = r [i].numerator / r [i].denominator;
        r [i].remainder = r [i].numerator % r [i].denominator;
        r [i].value = 1.0 * r [i].numerator / r [i].denominator;
    }

    cout << "Array was processed" << endl;
}

void arrayToOutfile (RationalNo r [], int size,
                    ofstream& outfile, char fileName [])
{
    outfile.open (fileName);

    if (!outfile)
    {
        cout << fileName
            << " opened for array transfer failed"
            << endl;

        exit (-1);
    }

    cout << "Begin array to outfile" << endl;

    outfile << fixed << showpoint << setprecision (4);

    for (int i = 0; i < size; i++)
    {
        outfile << r [i].numerator << "\t"
            << r [i].denominator << "\t"
            << r [i].quotient << "\t"
            << r [i].remainder << "\t"
            << r [i].value
            << endl;
    }

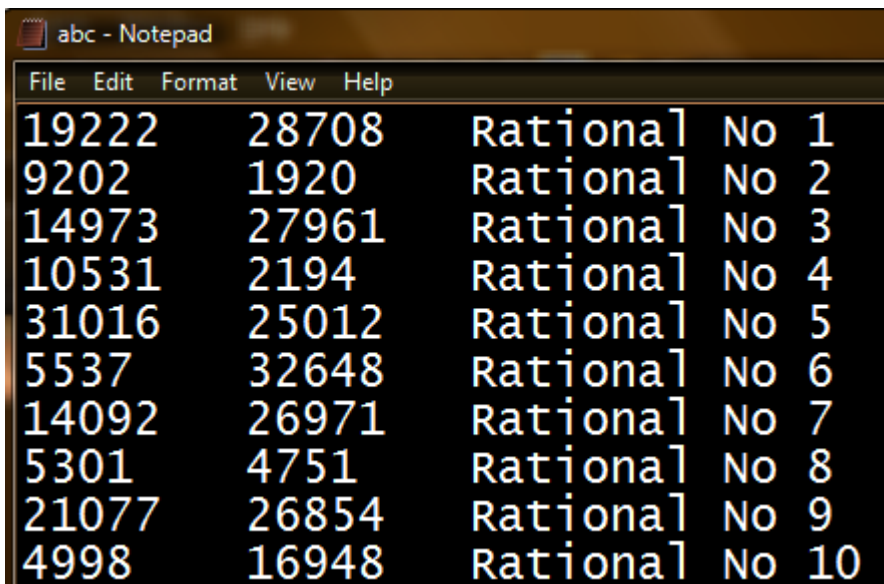
    outfile.close ();
}

```

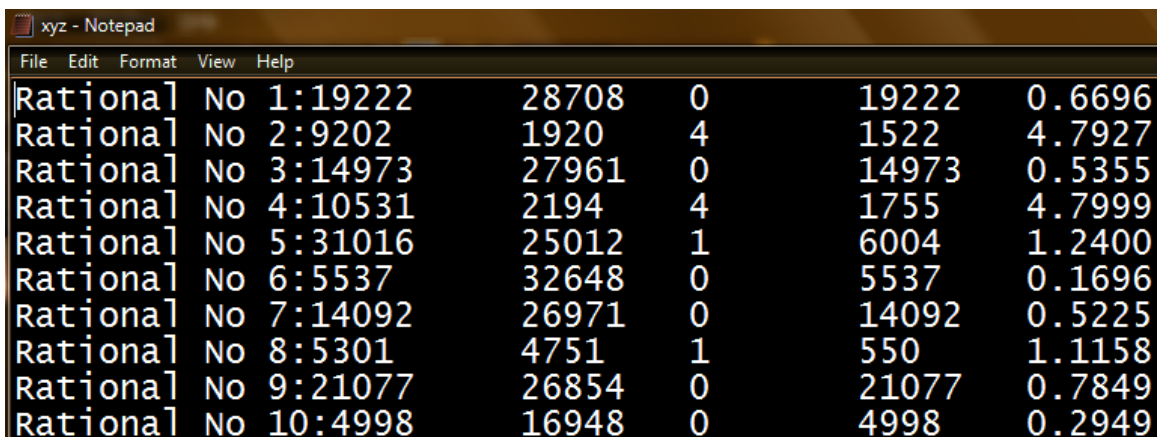
```

    cout << fileName << " generated" << endl;
}

```



19222	28708	Rational	No	1
9202	1920	Rational	No	2
14973	27961	Rational	No	3
10531	2194	Rational	No	4
31016	25012	Rational	No	5
5537	32648	Rational	No	6
14092	26971	Rational	No	7
5301	4751	Rational	No	8
21077	26854	Rational	No	9
4998	16948	Rational	No	10



Rational	No	1:19222	28708	0	19222	0.6696
Rational	No	2:9202	1920	4	1522	4.7927
Rational	No	3:14973	27961	0	14973	0.5355
Rational	No	4:10531	2194	4	1755	4.7999
Rational	No	5:31016	25012	1	6004	1.2400
Rational	No	6:5537	32648	0	5537	0.1696
Rational	No	7:14092	26971	0	14092	0.5225
Rational	No	8:5301	4751	1	550	1.1158
Rational	No	9:21077	26854	0	21077	0.7849
Rational	No	10:4998	16948	0	4998	0.2949

```

// file processing and array of structures
//
// 1. Create a data file not more than MAX, to
//    describe some structure definition
// 2. Transfer information from 1 to an array of
//    structures
// 3. Processing the array in 2
// 4. From array of structures to an output file
// 5. A main function to drive the task

```

```

#include <iostream>
#include <fstream>
#include <cstdlib>
#include <ctime>
#include <iomanip>

```

```

using namespace std;

const int MAX = 100;
const int MAXSTR = 80;

struct RationalNo
{
    int numer;
    int denom;
    int quotient;
    int remainder;
    double value;
    char label [MAXSTR];
};

// Make sure that the numerator has no zero digit int it
int getNoZeroDigitInt ();
bool noZero (int);
void constructInfile (fstream&, const char []);

int fileToArray (fstream&, const char [], RationalNo []);
void processArray (RationalNo [], int);
void arrayToOutfile (const RationalNo [], int, ofstream&, const char []);

int main ()
{
    fstream afile;
    char fileName [MAX];

    srand (time (NULL));

    cout << "Enter file name to be created: ";
    cin >> fileName;

    constructInfile (afile, fileName);

    cout << "-----" << endl;

    RationalNo r [MAX];

    int size = fileToArray (afile, fileName, r);

    cout << "-----" << endl;

```

```

    processArray (r, size);

    cout << "-----" << endl;

    ofstream outfile;

    cout << "Enter output file name: ";
    cin >> fileName;

    arrayToOutfile (r, size, outfile, fileName);
}

bool noZero (int n)
{
    while (n > 0)
    {
        if (n % 10 == 0)
            return false;

        n = n / 10;
    }

    return true;
}

int getNoZeroDigitInt ()
{
    int n;

    do
    {
        n = rand ();

    } while (noZero (n) == false);

    return n;
}

void constructInfile (fstream& afile, const char fileName [])
{
    afile.open (fileName, ios::out);
}

```

```

if (!afile)
{
    cout << fileName << " opened for creation failed" << endl;
    exit (-1);
}

cout << "Begin the creation of " << fileName << endl;

int size = rand () % 51 + 50;

for (int i = 1; i <= size; i++)
{
    afile << getNoZeroDigitInt () << "\t"
        << rand () + 1 << "\t"
        << " Rational No " << i
        << endl;
}

afile.close ();
cout << fileName << " successfully created" << endl;
}

int fileToArray (fstream& afile, const char fileName [],
                RationalNo r [])
{
    afile.open (fileName, ios::in);

    if (!afile)
    {
        cout << fileName << " opened for reading failed" << endl;
        exit (-1);
    }

    cout << "Begin from " << fileName << " to array" << endl;

    int i = 0;

    char tabKey;

    while (afile >> r [i].numer >> r [i].denom)
    {
        afile.get (tabKey); // read and throw
    }
}

```

```

        afile.getline (r [i].label, MAXSTR);

        ++i;
    }

    afile.close ();
    cout << "All records transferred to array" << endl;

    return i;
}

void processArray (RationalNo r [], int size)
{
    cout << "Begin the process of array" << endl;

    for (int i = 0; i < size; i++)
    {
        r [i].quotient = r [i].numer / r [i].denom;
        r [i].remainder = r [i].numer % r [i].denom;
        r [i].value = 1.0 * r [i].numer / r [i].denom;
    }

    cout << "Array was processed" << endl;
}

void arrayToOutfile (const RationalNo r [], int size,
                    ofstream& outfile, const char fileName [])
{
    outfile.open (fileName);

    if (!outfile)
    {
        cout << "Array to outfile failed" << endl;
        exit (-1);
    }

    cout << "Begin from array to " << fileName << endl;

    outfile << fixed << showpoint << setprecision (5);

    for (int i = 0; i < size; i++)
    {
        outfile << r [i].label << ": "

```

```

        << r [i].numer << "\t"
        << r [i].denom << "\t"
        << r [i].quotient << "\t"
        << r [i].remainder << "\t"
        << r [i].value
        << endl;
    }

    outfile.close ();

    cout << "Array to " << fileName << " OK" << endl;
}

```

// File processing + array of structures

//

// Three types of triangles

// - equilateral

// - isosceles

// - scalene

// Basic features

// - construct the input file

// - transfer input file to an array of structures

// - process the array

// - from array to output file

```
#include <iostream>
```

```
#include <fstream>
```

```
#include <cstdlib>
```

```
#include <ctime>
```

```
#include <cmath>
```

```
#include <cstring>
```

```
#include <iomanip>
```

```
using namespace std;
```

```
// Maximum size of input file
```

```
const int MAXSIZE = 100;
```

```
// Length of cstring
```

```
const int MAX = 80;
```

```
enum PossibleType {Equilateral, Isosceles, Scalene, NotTriangle};
```

```

struct Triangle
{
    int a;
    int b;
    int c;
    int perimeter;
    double area;
    PossibleType pt;
};

```

// To generate an input file

```

void get3Integers (int&, int&, int&);
void constructInfile (fstream&, char []);

```

// From input file to array and return

// the size of the array

```

int fileToArray (fstream&, char [], Triangle []);

```

// Process the array

```

bool testTriangle (int, int, int);
int getPerimeter (int, int, int);
double getArea (int, int, int);
PossibleType getType (int, int, int);
void processArray (Triangle [], int);

```

// Array to outfile

```

void getPTString (PossibleType, char []);
void arrayToOutfile (Triangle [], int, ofstream&, char []);

```

```

int main ()
{
    fstream afile;
    char fileName [MAX];

    srand (time (NULL));

    cout << "Enter the file name to be created: ";
    cin >> fileName;

    constructInfile (afile, fileName);

    cout << "-----" << endl;
}

```

```

Triangle t [MAXSIZE];

int size = fileToArray (afile, fileName, t);

cout << "-----" << endl;

processArray (t, size);

cout << "-----" << endl;

ofstream outfile;

cout << "Enter the output file name: ";
cin >> fileName;

arrayToOutfile (t, size, outfile, fileName);
}

void get3Integers (int& a, int& b, int& c)
{
    a = rand () % 5;
    b = rand () % 5;
    c = rand () % 5;
}

void constructInfile (fstream& afile, char fileName [])
{
    afile.open (fileName, ios::out);

    if (!afile)
    {
        cout << fileName << " opened for creation failed" << endl;
        exit (-1);
    }

    cout << "Begin the creation of " << fileName << endl;

    int a, b, c;

    int size = rand () % 51 + 50;

    for (int i = 1; i <= size; i++)

```

```

{
    get3Integers (a, b, c);

    afile << a << "\t"
        << b << "\t"
        << c << "\t"
        << "// possible triangle "
        << i
        << endl;
}

afile.close ();
cout << fileName << " successfully created" << endl;
}

int fileToArray (fstream& afile, char fileName [], Triangle t [])
{
    afile.open (fileName, ios::in);

    if (!afile)
    {
        cout << fileName << " opened for reading failed" << endl;
        exit (-1);
    }

    cout << "Begin from " << fileName
        << " to array"
        << endl;

    int i = 0;

    while (afile >> t [i].a >> t [i].b >> t [i].c)
    {
        afile.clear ();
        afile.ignore (MAX, '\n');

        ++i;
    }

    afile.close ();
    cout << fileName << " to array of triangles OK" << endl;

    return i;
}

```

```
}
```

```
bool testTriangle (int a, int b, int c)
{
    if (a + b > c && b + c > a && c + a > b)
        return true;
    else
        return false;
}
```

```
int getPerimeter (int a, int b, int c)
{
    return a + b + c;
}
```

```
double getArea (int a, int b, int c)
{
    double s = (a + b + c) / 2.0;

    return sqrt (s * (s - a) * (s - b) * (s - c));
}
```

```
PossibleType getType (int a, int b, int c)
{
    if (!testTriangle (a, b, c))
        return NotTriangle;
    else if (a == b && b == c)
        return Equilateral;
    else if (a == b || b == c || a == c)
        return Isosceles;
    else
        return Scalene;
}
```

```
void processArray (Triangle t [], int size)
{
    cout << "Begin the process of array" << endl;

    for (int i = 0; i < size; i++)
    {
        t [i].pt = getType (t [i].a, t [i].b, t [i].c);

        if (t [i].pt != NotTriangle)
```

```

        {
            t [i].perimeter = getPerimeter (t [i].a, t [i].b, t [i].c);
            t [i].area = getArea (t [i].a, t [i].b, t [i].c);
        }

    }

    cout << "Array was processed" << endl;
}

void arrayToOutfile (Triangle t [], int size,
                    ofstream& outfile, char fileName [])
{
    outfile.open (fileName);

    if (!outfile)
    {
        cout << fileName << " opened for creation failed" << endl;
        exit (-1);
    }

    cout << "Begin from array to outfile" << endl;

    char str [MAX];

    outfile << fixed << showpoint << setprecision (5);

    for (int i = 0; i < size; i++)
    {
        outfile << t [i].a << "\t"
                << t [i].b << "\t"
                << t [i].c << "\t";

        if (t [i].pt != NotTriangle)
        {
            outfile << t [i].perimeter << "\t"
                    << t [i].area << "\t";
        }
        else
            outfile << "N.A." << "\t" << "N.A.";

        getPTString (t [i].pt, str);
    }
}

```

```

        outfile << str
                << endl;
    }

    outfile.close ();
    cout << "Array to " << fileName << " OK" << endl;
}

void getPTString (PossibleType pt, char str [])
{
    switch (pt)
    {
        case Equilateral: strcpy (str, "Equilateral");
                        break;
        case Isosceles : strcpy (str, "Isosceles");
                        break;
        case Scalene : strcpy (str, "Scalene");
                        break;
        default : strcpy (str, "Not triangle");
    }
}

```

```

// File processing + array of structures
//
// Main objective
// - generate some strings of uppercase / lowercase / digits
// - extract three substrings
//   - one for lowercase
//   - one for uppercase
//   - one for digits
// - other properties
//
// Basic features
// - construct the input file
// - transfer input file to an array of structures
// - process the array
// - from array to output file

```

```

#include <iostream>
#include <fstream>
#include <cstdlib>
#include <ctime>
#include <cmath>
#include <cstring>
#include <cctype>
using namespace std;

```

```

// Maximum size of input file
const int MAXSIZE = 100;

```

```

// Length of cstring
const int MAX = 20;

```

```

struct StringInfo
{
    char str [MAX + 1];
    char lowerStr [MAX + 1];
    char upperStr [MAX + 1];
    char digitStr [MAX + 1];

    char label [MAX + 1];
};

```

// To generate an input file

```
char getUpperChar ();
char getLowerChar ();
char getDigitChar ();
void getString (char [], int);
void constructInfile (fstream&, const char []);
```

// From input file to array and return

// the size of the array

```
int fileToArray (fstream&, const char [], StringInfo []);
```

// Process the array

```
void getSubstrings (const char [], char [], char [], char []);
void processArray (StringInfo [], int);
```

// Array to outfile

```
void arrayToOutfile (const StringInfo [], int, ofstream&, const char []);
```

```
int main ()
{
    fstream afile;
    char fileName [MAX];

    srand (time (NULL));

    cout << "Enter the file name to be created: ";
    cin >> fileName;

    constructInfile (afile, fileName);

    cout << "-----" << endl;

    StringInfo t [MAXSIZE];

    int size = fileToArray (afile, fileName, t);

    cout << "-----" << endl;

    processArray (t, size);

    cout << "-----" << endl;

    ofstream outfile;
```

```

    cout << "Enter the output file name: ";
    cin >> fileName;

    arrayToOutfile (t, size, outfile, fileName);
}

char getUpperChar ()
{
    int k = rand () % 26 + 65;
    char ch = static_cast <char> (k);

    return ch;
}

char getLowerChar ()
{
    char lowercase [ ] = "abcdefghijklmnopqrstuvwxyz";

    int k = rand () % 26;

    return lowercase [k];
}

char getDigitChar ()
{
    char digit [] = "0123456789";

    int k = rand () % 10;
    return digit [k];
}

void getString (char str [], int size)
{
    int k;

    for (int i = 0; i < size; i++)
    {
        k = rand () % 3;

        switch (k)
        {
            case 0: str [i] = getUpperChar ();

```

```

        break;
    case 1: str [i] = getLowerChar ();
        break;
    case 2: str [i] = getDigitChar ();
    }
}

str [size] = '\0';
}

```

```

void constructInfile (fstream& afile, const char fileName [])
{
    afile.open (fileName, ios::out);

    if (!afile)
    {
        cout << fileName << " opened for creation failed" << endl;
        exit (-1);
    }

    cout << "Begin the creation of " << fileName << endl;

    int size = rand () % 51 + 50;

    char str [MAX + 1];

    for (int i = 1; i <= size; i++)
    {
        getString (str, MAX);

        afile << str
            << " String Info "
            << i
            << endl;
    }

    afile.close ();
    cout << fileName << " successfully created" << endl;
}

```

```

int fileToArray (fstream& afile, const char fileName [], StringInfo t [])

```

```

{
    afile.open (fileName, ios::in);

    if (!afile)
    {
        cout << fileName << " opened for reading failed" << endl;
        exit (-1);
    }

    int i = 0;
    char aSpace;

    while (afile >> t [i].str)
    {
        afile.get (aSpace);
        afile.getline (t [i].label, MAX + 1);

        ++i;
    }

    cout << "Begin from " << fileName
         << " to array"
         << endl;

    afile.close ();
    cout << fileName << " to array of StringInfos OK" << endl;

    return i;
}

```

```

void getSubstrings (const char str [],
                    char upperStr [],
                    char lowerStr [],
                    char digitStr [])
{
    int i = 0;
    int iU = 0;
    int iL = 0;
    int iD = 0;

    while (str [i] != '\0')
    {

```

```

        if (isupper (str [i]))
        {
            upperStr [iU] = str [i];
            ++iU;
        }
        else if (islower (str [i]))
        {
            lowerStr [iL] = str [i];
            ++iL;
        }
        else
        {
            digitStr [iD] = str [i];
            ++iD;
        }

        ++i;
    }

    upperStr [iU] = '\0';
    lowerStr [iL] = '\0';
    digitStr [iD] = '\0';

}

void processArray (StringInfo t [], int size)
{
    cout << "Begin the process of array" << endl;

    for (int i = 0; i < size; i++)
    {
        getSubstrings (t [i].str,
                        t [i].upperStr,
                        t [i].lowerStr,
                        t [i].digitStr);

    }

    cout << "Array was processed" << endl;
}

void arrayToOutfile (const StringInfo t [], int size,
                    ofstream& outfile, const char fileName [])

```

```

{
    outfile.open (fileName);

    if (!outfile)
    {
        cout << fileName << " opened for creation failed" << endl;
        exit (-1);
    }

    cout << "Begin from array to outfile" << endl;

    for (int i = 0; i < size; i++)
    {
        outfile << t [i].label << ": "
                << t [i].str << "\t"
                << t [i].upperStr << "\t"
                << t [i].lowerStr << "\t"
                << t [i].digitStr
                << endl;
    }

    outfile.close ();
    cout << "Array to " << fileName << " OK" << endl;
}

```

Revision Examples I

1. Number representation and operation

(a) Working on 24 bits representation, what is the 2's complement of 137 represented in octal? and in hexadecimal?

step 1

convert 137 to binary

$137 / 2 = 68 \text{ R } 1$
 $68 / 2 = 34 \text{ R } 0$
 $34 / 2 = 17 \text{ R } 0$
 $17 / 2 = 8 \text{ R } 1$
 $8 / 2 = 4 \text{ R } 0$
 $4 / 2 = 2 \text{ R } 0$
 $2 / 2 = 1 \text{ R } 0$
 $1 / 2 = 0 \text{ R } 1$

binary representation is
1000 1001

step 2

Internal representation
0000 0000 0000 0000 1000 1001

step 3 - 1's complement
1111 1111 1111 1111 0111 0110

step 4 - 2's complement, add 1 to step 3
1111 1111 1111 1111 0111 0111
F F F F 7 7 (in hexadecimal)

111 111 111 111 111 101 110 111
7 7 7 7 7 5 6 7 (in octal)

Operation (+ -)

=====

Compute the octal expression $534 - 276 = ???$, results in octal, in binary, in hexa and also in decimal.

--

5 3 4

2 7 6 -

2 3 6 (octal)

= 010011110 (binary)

= 0 1001 1110

= 9 E (hexadecimal)

= $2 * 8 * 8 + 3 * 8 + 6$

= $128 + 24 + 6 = 158$ (decimal)

9E to decimal

= $9 * 16 + 14$

= $144 + 14 = 158$ (decimal)

calculation

$8 + 4 - 6 = 6$

$8 + 2 - 7 = 3$

Sequential design

data types
variables
assignment statements
increment ++ / decrement operators --
cin / cout
clear / ignore
formatted output (setw, setfill, precision)

Sample example,

(a) What is the output of the following code fragment if we enter the following input?

1234.4564 678.3427 ABCDEFG

```
char ch1, ch2;  
int m, n;  
float x, y;  
  
cin >> ch1;           // ch1 = '1'  
cin >> m >> x;         // m = 234, x = 0.4564  
cin.get(ch2);         // ch2 = '  
cin >> n >> y;         // n = 678, y = 0.3427  
  
cout << fixed << showpoint << setprecision(3);  
cout << m << n << ch1 << ch2 << x << y << endl;
```

2346781 0.4560.343 (ANS)

(b) What is the output of the following code fragment if we enter the following input twice?

1234.4564 678.3427 ABCDEFG
1234.4564 678.3427 ABCDEFG

```
char ch1, ch2;  
int m, n;
```

```

float x, y;

cin >> ch1;           // ch1 = '1'
cin >> m >> x;         // m = 234, x = 0.4564
cin.clear ();
cin.ignore (100, '\n');
cin.get (ch2);         // ch2 = 1
cin >> n >> y;         // n = 234 , y = 0.4564

cout << fixed << showpoint << setprecision (3);
cout << m << n << ch1 << ch2 << x << y << endl;

```

Output is
234234110.4560.456

What is the output of the following code fragment if we enter the following input?

1234.4564 678.3427 ABCDEFG

```

char ch1, ch2;
int m, n;
float x;
int y;

cin >> ch1;           // ch1 = '1'
cin >> m >> x;         // m = 234, x = 0.4564
cin.get (ch2);        // ch2 = ' '
cin >> n >> y;         // n = 678, y = ??

cout << fixed << showpoint << setprecision (3);
cout << m << n << ch1 << ch2 << x << y << endl;

```

Answer: run time error

What is the output of the following code fragment if we enter the following input?

1234.4564 678.3427 ABCDEFG

```

char ch1, ch2;

```

```

int m, n;
float x;
int y;

cin >> ch1;           // ch1 = '1'
cin >> m >> x;         // m = 234, x = 0.4564
cin.get (ch2);         // ch2 = ' '
cin >> n >> ch2 << y;   // n = 678, ch2 = '.'
                        // y = 3427

cout << fixed << showpoint << setprecision (3);
cout << m << n << ch1 << ch2 << x << y << endl;

```

output is

2346781.0.4563427

What is the output of the following code fragment if we enter the following input?

1234.4564 678.3427 ABCDEFG

```

char ch1, ch2;
int m, n;
float x;
int y;

cin >> ch1;           // ch1 = '1'
cin >> m >> x;         // m = 234, x = 0.4564
cin.get (ch2);         // ch2 = ' '
cin >> n >> ch2 << y;   // n = 678, ch2 = '.'
                        // y = 3427

cout << fixed << showpoint << setprecision (3);
cout << m-- << n++ << ch1 << ch2 << x << y << n << m << endl;

```

output is

2346781.0.4563427679233

=====

Selection design

if if-else switch-case statements

(a) What is the output of the following code fragment?

```
int x = 1, y = 2, z = 3;

if ((x = 2) || (y = 3) || (z = 4))
{
    x += 2;      // 4
    y += 3; // 5
    z += 4; // 7
}
else
    x += 10;

cout << x << y << z << endl;
```

```
int x = 1, y = 2, z = 3;

if ((x == 2) || (y == 3) || (z = 4))
{
    x += 2;      // 3
    y += 3; // 4
    z += 4; // 8
}
else
    x += 10;

cout << x << y << z << endl;
```

```
int x = 1, y = 2, z = 3;

if ((x == 2) || (y == 3) || (z = 0))
{
    x += 2;      //
    y += 3; //
```

```

        z += 4; //
    }
    else
        x += 10;

cout << x << y << z << endl;

```

Answer: 1120

What is the output of the following code fragment?

```

int x = 1, y = 2, z = 3;

if ((x = 2) && (y = 3) && (z = 4))
{
    x += 2;      // 4
    y += 3; // 6
    z += 4; // 8
}
else
    x += 10;

cout << x << y << z << endl;

```

What is the output of the following code fragment?

```

int x = 1, y = 2, z = 3;

if ((x = 2) && (y = 3) && (z == 4))
{
    x += 2;
    y += 3;
    z += 4;
}
else
    x += 10;

cout << x << y << z << endl;

```

output 1133

What is the output of the following code fragment?

```
int x = 1, y = 2, z = 3;

if ((x = 2) && (y = 3) && (z = 4))
{
    x += 2;      // 4
    y += 3; // 6
    z += 4; // 8
}
else
    x += 10;

cout << x << y << z << endl;
```

output: 468

```
int x = 6;

if (x + 5 = 7)
    blah blah blah;
```

syntax error

Repetition designs

=====

while
for
do-while
nested loops

Functions

=====

1. How to design a function?

- name of the function
- what information is important to the function?
i.e the parameters
- what information to be returned?
i.e. return type

(a) Design a function to draw a line of k symbols.

- drawALine
- parameters: k (integer), symbol (character)
- void

function prototype is

`void drawALine (int, char);`

```
void drawALine (int k, char symbol)
{
    for (int i = 1; i <= k; i++)
        cout << symbol;
    cout << endl;
}
```

(b) Design a function to draw n lines of k symbols

- drawNLines
- parameters: n, k, symbol
- void

```
void drawNLines (int, int, char);

void drawNLines (int n, int k, char symbol)
{
    for (int i = 1; i <= n; i++)
        drawALine (k, symbol);
}
```

(c) An integer is symmetric if and only if you read the integer from left to right and from right to left they are the same. For example, 12321, 1221 are symmetric and 123421 is not symmetric.

Design a few functions to achieve the target.

What algorithm?

Algorithm 1
=====

read in number and reverse it and see if the two numbers are the same, if so ==> symmetric

symmetric_1.cpp - explore passing by value

```
#include <iostream>
using namespace std;

int getReverse (int);

int main ()
{
    int n;

    cout << "Enter a number: ";
    cin >> n;

    int reverse = getReverse (n);

    if (reverse == n)
        cout << n << " is symmetric" << endl;
    else
        cout << n << " is not symmetric" << endl;
```

```

}

int getReverse (int n)
{
    int reverse = 0;
    while (n > 0)
    {
        reverse = reverse * 10 + n % 10;
        n /= 10;
    }

    return reverse;
}

```

symmetric_2.cpp - explore passing by reference

```

// same as symmetric_1.cpp
// explore the concept of passing by reference

#include <iostream>
using namespace std;

void getReverse (int, int&);

int main ()
{
    int n,reverse;

    cout << "Enter a number: ";
    cin >> n;

    getReverse (n, reverse);

    if (reverse == n)
        cout << n << " is symmetric" << endl;
    else
        cout << n << " is not symmetric" << endl;
}

void getReverse (int n, int& reverse)
{

```

```

reverse = 0;

while (n > 0)
{
    reverse = reverse * 10 + n % 10;
    n /= 10;
}

}

```

Algorithm 2

=====

Given a number

- get the last digit
- get the first digit
- if (first digit != last digit)
 - return not symmetric
- eliminate last digit
- eliminate 1st digit
- and repeat until the number has less than or equal to 1 digit, then return true

see program symmetric_3

```

#include <iostream>
using namespace std;

```

```

bool isSymmetric (int);

```

```

int getLastDigit (int);
int getFirstDigit (int);
int removeLastDigit (int);
int removeFirstDigit (int);

```

```

int main ()
{
    int n;

```

```

    cout << "Enter a positive integer: ";
    cin >> n;

    if (isSymmetric (n))
        cout << n << " is symmetric" << endl;
    else
        cout << n << " is not symmetric" << endl;
}

bool isSymmetric (int n)
{
    int firstDigit, lastDigit;

    while (n > 10)
    {
        lastDigit = getLastDigit (n);
        firstDigit = getFirstDigit (n);

        if (firstDigit != lastDigit)
            return false;

        n = removeFirstDigit (n);
        n = removeLastDigit (n);
    }

    return true;
}

int getLastDigit (int n)
{
    return n % 10;
}

int getFirstDigit (int n)
{
    while (n > 10)
        n /= 10;

    return n;
}

int removeLastDigit (int n)
{

```

```

        n = n / 10;
        return n;
    }

int removeFirstDigit (int n)
{
    // form a reverse without the 1st digit
    int reverse = 0;
    while (n > 10)
    {
        reverse = reverse * 10 + n % 10;
        n /= 10;
    }

    // reverse back
    n = 0;
    while (reverse > 0)
    {
        n = n * 10 + reverse % 10;
        reverse = reverse / 10;
    }

    return n;
}

```

```

// same as symmetric_3.cpp
//
// explore passing by reference
#include <iostream>
using namespace std;

```

```

bool isSymmetric (int);

```

```

void getLastDigit (int, int&);
void getFirstDigit (int, int&);
void removeLastDigit (int&);
void removeFirstDigit (int&);

```

```

int main ()
{
    int n;

    cout << "Enter a positive integer: ";

```

```

    cin >> n;

    if (isSymmetric (n))
        cout << n << " is symmetric" << endl;
    else
        cout << n << " is not symmetric" << endl;
}

bool isSymmetric (int n)
{
    int firstDigit, lastDigit;

    while (n > 10)
    {
        getLastDigit (n, lastDigit);
        getFirstDigit (n, firstDigit);

        if (firstDigit != lastDigit)
            return false;

        removeFirstDigit (n);
        removeLastDigit (n);
    }

    return true;
}

void getLastDigit (int n, int& lastDigit)
{
    lastDigit = n % 10;
}

void getFirstDigit (int n, int& firstDigit)
{
    while (n > 10)
        n /= 10;

    firstDigit = n;
}

void removeLastDigit (int& n)
{
    n = n / 10;
}

```

```
}
```

```
void removeFirstDigit (int& n)
{
    // form a reverse without the 1st digit
    int reverse = 0;
    while (n > 10)
    {
        reverse = reverse * 10 + n % 10;
        n /= 10;
    }

    // reverse back
    n = 0;
    while (reverse > 0)
    {
        n = n * 10 + reverse % 10;
        reverse = reverse / 10;
    }
}
```

```
// same as symmetric_3.cpp
// a recursive function
//
// base case:
// - all single digits, i.e. < 10 is symmetric
// general case;
// - out initial algorithm
```

```
#include <iostream>
using namespace std;
```

```
bool isSymmetric (int);
```

```
int getLastDigit (int);
int getFirstDigit (int);
int removeLastDigit (int);
int removeFirstDigit (int);
```

```
int main ()
{
```

```

int n;

cout << "Enter a positive integer: ";
cin >> n;

if (isSymmetric (n))
    cout << n << " is symmetric" << endl;
else
    cout << n << " is not symmetric" << endl;
}

bool isSymmetric (int n)
{
    if (n < 10)
        return true;
    else if (getFirstDigit (n) != getLastDigit (n))
        return false;
    else
    {
        n = removeLastDigit (n);
        n = removeFirstDigit (n);
        return isSymmetric (n);
    }
}

int getLastDigit (int n)
{
    return n % 10;
}

int getFirstDigit (int n)
{
    while (n > 10)
        n /= 10;

    return n;
}

int removeLastDigit (int n)
{
    n = n / 10;
    return n;
}

```

```

int removeFirstDigit (int n)
{
    // form a reverse without the 1st digit
    int reverse = 0;
    while (n > 10)
    {
        reverse = reverse * 10 + n % 10;
        n /= 10;
    }

    // reverse back
    n = 0;
    while (reverse > 0)
    {
        n = n * 10 + reverse % 10;
        reverse = reverse / 10;
    }

    return n;
}

```

Algorithm 3 – Using arrays

// using array to check if an integer is symmetric

```
#include <iostream>
```

```
using namespace std;
```

```
const int MAX = 10;
```

```
bool isSymmetric (int);
```

```
int transferIntToArray (int, int []);
```

```
bool arraySymmetric (const int [], int);
```

```
int main ()
```

```
{
    int n;
```

```
    cout << "Enter a positive integer: ";
    cin >> n;
```

```
    if (isSymmetric (n))
```

```

        cout << n << " is symmetric" << endl;
    else
        cout << n << " is not symmetric" << endl;
}

bool isSymmetric (int n)
{
    int a [MAX];

    int size = transferIntToArray (n, a);

    return arraySymmetric (a, size);
}

int transferIntToArray (int n, int a [])
{
    int i = 0;

    while (n > 0)
    {
        a [i] = n % 10;
        n = n / 10;
        ++i;
    }

    return i;
}

bool arraySymmetric (const int a [], int size)
{
    int left = 0;
    int right = size - 1;

    while (left < right)
    {
        if (a [left] != a [right])
            return false;

        ++left;
        --right;
    }

    return true;
}

```

}

```

// using array to check if an integer is symmetric
//
// same as symmetric_6.cpp, recursive design
#include <iostream>
using namespace std;

const int MAX = 10;

bool isSymmetric (int);
int transferIntToArray (int, int []);
bool arraySymmetric (const int [], int left, int right);

int main ()
{
    int n;

    cout << "Enter a positive integer: ";
    cin >> n;

    if (isSymmetric (n))
        cout << n << " is symmetric" << endl;
    else
        cout << n << " is not symmetric" << endl;
}

bool isSymmetric (int n)
{
    int a [MAX];

    int size = transferIntToArray (n, a);

    return arraySymmetric (a, 0, size - 1);
}

int transferIntToArray (int n, int a [])
{
    int i = 0;

    while (n > 0)
    {
        a [i] = n % 10;
        n = n / 10;
        ++i;
    }
}

```

```
    }  
    return i;  
}  
  
bool arraySymmetric (const int a [], int left, int right)  
{  
    if (left >= right)  
        return true;  
    else if (a [left] != a [right])  
        return false;  
    else  
        return arraySymmetric (a, ++left, --right);  
}
```

Revision Examples II

How to trace the result of a recursive function?

For example

```
int function (int m, int n)
{
    if (n == 0 && m == 0)
        return 10;
    else if (n != 0 && m == 0)
        return 2 * n;
    else if (n == 0 && m != 0)
        return 3 * m;
    else
        return (m % 10 + n % 10) + function (m / 10, n / 10);
}
```

what is the output

```
cout << function (1234, 5678) << endl;
cout << function (123456, 5678) << endl;
cout << function (1234, 567892) << endl;
```

Answers

=====

```
function (1234, 5678)
= (4 + 8) + fuction (123, 567)
= 12 + (3 + 7) + function (12, 56)
= 22 + (2 + 6) + function (1, 5)
= 30 + (1 + 5) + function (0, 0)
= 36 + 10
= 46
```

```
function(123456,5678)
=(6+8)+function(12345,567)
=14+(5+7)+function(1234,56)
=26+(4+6)+function(123,5)
=36+(3+5)+function(12,0)
=44+3*12
=44+36
```

=80

```
function (1234, 567892)
= (4 + 2) + function (123, 56789)
= 6 + (3 + 9) + function (12, 5678)
= 18 + (2 + 8) + function (1, 567)
= 28 + (1 + 7) + function (0, 56)
= 36 + 2 * 56
= 36 + 112
= 148
```

Design a function with the following prototype:

```
int function (int, int, char);
```

Before you implement, fill in the parameters:

```
int function (int m, int n, char ch)
{

    return ...;
}
```

```
while ( ** && **)
```

```
while (A > 0 && B != 0)
{
```

Inside the while loop, A eventually will be ≤ 0 OR
B will be equal to 0 in order to exit from the while loop

```
}
```

```
int A = 123;
int B = 10;
```

```
while (A > 0 && B != 0)
{
    cout << A % 10 << B;
```

```

    A /= 10;
    B = B - 5;
}

```

```
cout << A << B << endl;
```

A	B	condition	cout << ...
123	10	true	3102510
12	5	true	
1	0	false	

```

while ( ** || ** )
{

}

```

```

int A = 123;
int B = 10;

```

```

while (A > 0 || B != 0)
{
    cout << A % 10 << B;
    A /= 10;
    B = B - 5;
}

```

A	B	condition	cout <<
123	10	A > 0? yes	31025100-5
12	5	A > 0? yes	
1	0	A > 0? yes	
0	-5	A > 0? no	
		B != 0 yes	
0	-10	etc infinite loop	

```
while (! (** && **))
```

equal to

```
while (** || **)
```

while (! (** || **))

equals to

while (** && **)

Assignment 3 Q14

```
int countDigits (int noOfChars)
{
    int count = 0;
    char ch;

    for (int i = 1; i <= noOfChars; i++)
    {
        cout << "Enter a character: ";
        cin >> ch;

        if (isdigit (ch))
            ++count;

    }

    return count;
}
```

```
int countDigits (int noOfChars)
{
    int count = 0;
    char ch;

    for (int i = 1; i <= noOfChars; i++)
    {
        cout << "Enter a character: ";
        cin >> ch;

        if (ch >= '0' && ch <= '9')
            ++count;

    }
}
```

```

        return count;
    }

int countDigits (int noOfChars)
{
    int count = 0;
    char ch;

    for (int i = 1; i <= noOfChars; i++)
    {
        cout << "Enter a character: ";
        cin >> ch;

        switch (ch)
        {
            case '0':
            case '1':
            case '2':
            case '3':
            case '4':
            case '5':
            case '6':
            case '7':
            case '8':
            case '9': ++count;
        }
    }

    return count;
}

```

Description as above, try to find and to return the sum of digits characters. Digit character should be treated as its integer value rather than the ascii value, i.e. '9' should be treated as integer 9.

```

int sumDigits (int noOfChars)
{
    int sum = 0;
    char ch;

```

```

for (int i = 1; i <= noOfChars; i++)
{
    cout << "Enter a character: ";
    cin >> ch;

    if (ch >= '0' && ch <= '9')
        sum = sum + ch - '0';

}

return count;
}

```

```

int sumDigits (int noOfChars)
{
    int sum = 0;
    char ch;

    for (int i = 1; i <= noOfChars; i++)
    {
        cout << "Enter a character: ";
        cin >> ch;

        switch (ch)
        {
            case '0':
            case '1':
            case '2':
            case '3':
            case '4':
            case '5':
            case '6':
            case '7':
            case '8':
            case '9': sum += ch - '0';

        }

    }

    return count;
}

```

Boolean variable - bool type

=====

Design a function to check if a digit is inside an integer, and the function has the following prototype:

```
bool isIn (int, int);

bool isIn (int anInteger, int aDigit)
{
    bool found = false;

    while (!found && anInteger > 0 )
    {
        if (aDigit == n % 10)
            found = true;
        else
            anInteger = anInteger / 10;
    }

    return found;
}
```

Same question as above, using the following prototype:

```
void isIn (int anInteger, int aDigit, bool& found)
{
    found = false;

    while (!found && anInteger > 0 )
    {
        if (aDigit == n % 10)
            found = true;
        else
            anInteger = anInteger / 10;
    }
}
```

=====

What is a c-string?

- A c-string is an array of characters ended with '\0'

How to process a c-string str?

basic syntax;

```
int i = 0;
while (str [i] != '\0')
{
    ....
    ++i;
}
```

design a function tp construct randomly a string of uppercase letters with the following prototype:

void upperStr (char [], int);

A few designs

=====

// If you know the ascii values

void upperStr (char str [], int size)

```
{
    for (int i = 0; i < size - 1; i++)
        str [i] = static_cast <char> (rand () % 26 + 65);
    str [size - 1] = '\0';
}
```

// If you do not know the ascii values

void upperStr (char str [], int)

```
{
    char upperCase [] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

    for (int i = 0; i < size - 1; i++)
        str [i] = uppercase [rand () % 26];
    str [size - 1] = '\0';
}
```

```
}
```

Design a function to construct a string of letters.

```
void letterStr (char str [], int)
{
    char letter [] =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";

    for (int i = 0; i < size - 1; i++)
        str [i] = letter [rand () % 52];
    str [size - 1] = '\0';
}
```

```
void letterStr (char str [], int)
{
    char upperLetter [] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    char lowerLetter [] = "abcdefghijklmnopqrstuvwxyz";

    int dice;

    for (int i = 0; i < size - 1; i++)
    {
        dice = rand () % 2;

        switch (dice)
        {
            case 0:    str [i] = upperLetter [rand () % 26];
                       break;
            case 1: str [i] = lowerLetter [rand () % 26];
                       break;
            default: // future enhancement
        }
    }

    str [size - 1] = '\0';
}
```

Boolean type applied to c-string

=====

Design a function to check if a vowel is inside a c-string with the following prototype:

```
bool isVowelIn (const char []);
```

```
bool isVowelIn (const char str [])
{
    int i = 0;
    bool isIn = false;

    while (!isIn && str [i] != '\0')
    {
        if (isVowel (str [i]))
            isIn = true;
        else
            ++i;
    }

    return isIn;
}
```

Design a function to check if a character is a vowel.

```
bool isVowel (char ch)
{
    switch (ch)
    {
        case 'a':
        case 'A':
        case 'e':
        case 'E':
        case 'i':
        case 'I':
        case 'o':
        case 'O':
        case 'u':
        case 'U': return true;
        default : return false;
    }
}
```

=====

Design a function to test if a word is symmetric (case sensitive).
For example, ABBA, abba, madam, MADAM are symmetric
ABba, MadAm are not symmetric

The function prototype is

```
bool isSymmetric (const char []);
```

```
bool isSymmetric (const char word [])
{
    int leftIndex = 0
    int rightIndex = strlen (word) - 1;

    while (leftIndex < rightIndex)
    {
        if (word [leftIndex] != word [rightIndex])
            return false;

        ++leftIndex;
        --rightIndex;
    }

    return true;
}
```

what about non-case sensitive version
AbBa and MadAm are symmetric

```
bool isSymmetric (const char word [])
{
    int leftIndex = 0
    int rightIndex = strlen (word) - 1;

    while (leftIndex < rightIndex)
    {
        if (toupper (word [leftIndex]) != toupper (word [rightIndex]))
            return false;

        ++leftIndex;
        --rightIndex;
    }
}
```

```
    }  
    return true;  
}
```

Sample Questions

If the input is:

```
12.34.56.78    two lines of input
12.34.56.78
```

what will the output of the following C++ code fragment be?

```
char ch1, ch2;
int n;
float x;

cout << fixed << showpoint << setprecision (2);

cin >> n >> ch1;

cin.clear ();
cin.ignore (100, '\n');

cin.get (ch2);
cin >> x;

cout << ch1 << ch2 << n << x << endl;
```

- (A) 12.34.78**
- (B) .1234.560**
- (C) .1122.34**
- (D) .12.34.56.78**
- (E) None of the above**

What is the output of the following C++ code fragment?

```
int n = 200;
int m = 400;

if (n = 10)          // Yes, this is an equal sign
{
    if (n)
        n += m;
    else
        n *= m--;
}
else
    n = n + m;

cout << n << m << endl;
```

- (A) 200400
- (B) 410400
- (C) 600400
- (D) 400600
- (E) None of the above

What is the output of the following C++ code fragment?

```
int index = 3;
while (index > 0)
{
    int k = --index;
    while (k > 0)
    {
        cout << k;
        k -= 2;
    }
}

cout << endl;
```

- (A) 11**
- (B) 31**
- (C) 14**

- (D) 21**
- (E) None of the above**

Perform the following conversions or operations. Show all workings.

- (a) Derive the 2's complement of the decimal number -977 stored in a 24 bits representation in hexadecimal and in octal.

First convert the number to binary

$$\begin{array}{rcl} 977 / 2 & = & 488 \quad \text{R } 1 \\ 488 / 2 & = & 244 \quad \text{R } 0 \\ 244 / 2 & = & 122 \quad \text{R } 0 \\ 122 / 2 & = & 61 \quad \text{R } 0 \\ 61 / 2 & = & 30 \quad \text{R } 1 \\ 30 / 2 & = & 15 \quad \text{R } 0 \\ 15 / 2 & = & 7 \quad \text{R } 1 \\ 7 / 2 & = & 3 \quad \text{R } 1 \\ 3 / 2 & = & 1 \quad \text{R } 1 \end{array}$$

The binary 24 bits representation for $+977$ is

0000 0000 0000 0011 1101 0001

The One's complement is

1111 1111 1111 1100 0010 1110

The Two's complement is

1111 1111 1111 1100 0010 1111

F F F C 2 F (in hexadecimal)

111 111 111 111 110 000 101 111

7 7 7 7 6 0 5 7 (in octal)

- (b) What is the result of the octal expression $645_8 - 377_8$ in decimal?

$$\begin{array}{r} 645 \\ 377 - \\ \hline \end{array}$$

246 (in octal)

$$\begin{aligned} &= 2 * 8 * 8 + 4 * 8 + 6 \\ &= 128 + 32 + 6 \\ &= 166 \text{ (in decimal)} \end{aligned}$$

Given three integers x , y and z . We wish to know whether the 3rd integer z is greater or is smaller or is in between when comparing to x and y .

- (a) Define an enumeration data type called `Status` that has three enumeration constants `Greatest`, `Smallest` and `Between`.
- (b) Design a function to determine the status of the third parameters comparing to the 1st and 2nd parameters. The function had the following prototype:

```
void getStatus (int, int, int, Status&);
```

```
enum Status {Greatest, Smallest, Between};
```

```
void getStatus (int x, int y, int z, Status& s)
{
    if (z >= x && z >= y)
        s = Greatest;
    else if (z <= x && x <= y)
        s = Smallest;
    else
        s = Between;
}
```

The dot product of two vectors $\mathbf{a} = [a_1, a_2, \dots, a_n]$ and $\mathbf{b} = [b_1, b_2, \dots, b_n]$ is defined as

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + \dots + a_n b_n$$

As can be seen from the above notations, we can use two one-dimensional arrays to denote the vectors. Assume that $\mathbf{a}$ and $\mathbf{b}$ are two vectors and each consists of n integer elements. Compare the designs using two different approaches:

- (a) iterative and
- (b) recursive

to find the dot product of the two vectors using your preferable function prototypes.

```
int dotProduct (int a [ ], int b [ ], int n)
{
    int sumProducts = 0;

    for (int i = 0; i < n; i++)
        sumProducts += a [i] * b [i];

    return sumProducts;
}
```

```
int dotProductR (int a [ ], int b [ ], int n)
{
    if (n == 0)
        return 0;
    else
        return a [n - 1] * b [n - 1] + dotProductR (a, b, n - 1);
}
```

A C-string `str` (an array of characters) composes of some ASCII characters; some of the characters may be digit characters. Design a function to return (via the reference parameters) the number of odd character digits, the number of even character digits and the sum of all character digits (treated as their corresponding integer digits, not the ASCII values).

For example, `str = "2R4567UJK&&$9"`, the number of odd digits is 4, the number of even digits is 2, the sum of digits is 33 (which is $2 + 4 + 5 + 6 + 7 + 9$). The function has the following prototype:

```
void digitsInfo (const char [], int&, int&,
                int&);
```

```
void digitsInfo (const char str [ ], int& noOdds, int& noEven, int&
sumDigits)
{
    noOdds = 0;
    noEvens = 0;
    sumDigits = 0;
    int k = 0;

    int n;

    while (str [k] != '\0')
    {
        if (isdigit (str [i]))
        {
            n = str [i] - '\0';
            sumDigits += n;

            if (n % 2)
                noOdds++;
            else
                noEvens++;
        }

        ++k;
    }
}
```

Let A be a two dimensional array of real numbers with m rows and n column. Design a function to return, via the reference parameters, the largest value and the smallest value stored in A. The function has the following prototype:

```
void differenceMaxMin (const float [ ][MAX], int,  
                      int, float&, float&);  
// where MAX is constant indicating the maximum  
   number of columns
```

```
void differenceMaxMin (const float a [ ][MAX], int m, int n, int  
difference)  
{  
    float max = a [0][0];  
    float min = a [0][0];  
  
    for (int row = 0; row < m; row++)  
    {  
        for (int col = 0; col < n; col++)  
        {  
            if (max < a [row][col])  
                max = a [row][col];  
  
            if (min > a [row][col])  
                min = a [row][col];  
        }  
    }  
}
```

A non-empty text file contains only some ASCII characters . The following shows one of the sample text files:

```
A c d e H 7 8 5 % ^ n      K n 8 6 7
A b c T 6 %   7 4 ##### @ @ @ ) )
```

```
U I y nnn   3 2 4
```

Design a function to count and to return the number of uppercase letters and the number of lower case letters stored in this text file. Your function must have the following prototype:

```
void countUpperLower (const char fileName [ ],
                      int& noUppers, int&
noLowers) ;
```

```
ifstream infile;
infile.open (fileName);
```

```
char ch;
noUppers = 0;
noLowers = 0;
```

```
// Note that you need to use get function
// to get a character as it can be a space
while (infile.get (ch))
{
```

```
    if (isupper (ch))
        noUppers++;
    else if (islower (ch))
        noLowers++;
```

```
}
```

```
infile.close ();
```

```
// You can also replace the if else statement
// inside the while loop by
```

```
if (ch >= 'A' && ch <= 'Z')
    noUppers++;
else if (ch >= 'a' && ch <= 'z')
    ++noLowers;
```

*ALL
THE
BEST
TO
YOUR
STUDY*

Best wishes from Heng Aik Koan