**Operators**

We can use C/C++ to perform mathematical calculations like 5 + 6, 6 / 2 etc.
The basic symbols used to perform mathematical operations are called operators.
For example, consider the statement: c = a + b;
Here, '+' is the operator known as addition operator and 'a' and 'b' are operands.
Operators tell compiler what to do with the numbers surrounding them. The Plus sign(+) adds numbers, the minus sign (-) subtracts numbers, the asterisk (*) multiplies numbers, and the slash (/) divides numbers.

**Example in C++:**

```
#include <iostream>
using namespace std;
int main()
{
  cout<<(300 + 100);
}
```

Output of the above C++ code will be 400.

**Note**:

In the above program first line is to include header file for input and output. Generally, a program includes various programming elements like built-in functions, classes, keywords, constants, operators etc., that are already defined in the standard C++ library. In order to use such pre-defined elements in a program, an appropriate header must be included in the program.
In C++ we use **cout** to display the content and **cin** to take input from user while in C, we use **printf** to display the content and **scanf** to take input from user
The main() is a startup function that starts the execution of a C/C++ program. All C/C++ statements that need to be executed are written within main(). The compiler executes all the instructions written within the opening and closing curly braces' {}' that enclose the body of main(). Once all the instructions in main() are executed, the control passes out of main(), terminating the entire program and returning a value to the operating system.

**Question:** Write a C/C++ code to print the result of the following expressions in different lines:
1. Sum of 200 and 300
2. Division of 250 by 50
3. Difference of 500 and 100
4. Product of '**a**' and 50 ,where '**a**' is a keyboard input

```
#include<iostream>
using namespace std;
int main()
{
        cout<<(300+200)<<endl;
  cout<<(250/50)<<endl;
  cout<<(500-100)<<endl;
  int a;
  cin>>a;
  cout<<a*50;
  //write your code here

}
```

**Evaluating Expressions**

Expressions are made up of values (the numbers) connected by operators (the math signs) that produce a new value the code can use.

10 + 30 is an example of expression. Values 10 and 30 are connected by an operator ' + '.

When the computer solves the expression 10 + 30 and returns the value 40, it has evaluated the expression. Evaluating an expression reduces the expression to a single value, just like solving a math problem reduces the problem to a single number: the answer.

When a program evaluates an expression, it has an order of precedence in which these are going to be calculated, just like you do when you do math.

There are just a few rules:

1. Multiplication and division always go before addition and subtraction.
2. We use parentheses to control the order of operations. Parts of the expression inside parentheses are evaluated first.
3. Parentheses can be nested, which means that there can be parentheses inside parentheses.
4. Program evaluates the innermost parentheses first, then the outer ones.
5. If two operators have same precedence, then their associativity is checked. The operators +, -, * and / are left associative. This means that if we have the expression 3 * 8 / 3, then first 3 * 8 is evaluated, then 24 / 3 is evaluated.

**Example**:

The expression 7 + 6 * 2 + 4 evaluates to 23, not 30, because 6 * 2 is evaluated first. If the expression were (7 + 6) * (2 + 4) it would evaluate to 78, because the (7 + 6) and (2 + 4) inside parentheses are evaluated before multiplication.

**Question**: Write a program to print the result of the following statement

Add 10 to 8, then Multiply the result by 2, then subtract 11 from the result, and divide it by **'a'** where a is a keyboard input.

```
#include <iostream>
using namespace std;
int main()
{
 int a;
  cin>>a;
  cout<<(((10+8)*2)-11)/a;//write your code here
}
```

---

**Variables**

The word variable in programming describes a place to store information such as numbers, text, lists of numbers and so on.

When an expression evaluates to a value, you can use that value later by storing it in a variable. A variable has a data type which tell variable the type of data that can be stored in it.

**Type & Description:**

bool - Stores either true or false.

char - Smallest addressable unit of the machine that can contain basic character set.

int - Used to store integer values.

float - Used to store single-precision floating point values.

double - Used to store double-precision floating point values.

void - A void variable cannot be initialised with any value. This is usually used to specify the type of functions which return nothing.

**To assign a value to a variable :** type variable_name = value;

For example, `int price = 500;`

**Question**: Write a program to assign the value '**a**' (where '**a**' is a keyboard input) to the variable `first`, then assign the contents of first to variable `second`. Print the variable '`second`'

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

int main()
{
int a;
 cin>>a;
  int first=a;
  int second=first;
  cout<<second;
  //write your code here

}
```

**Naming a variable**

All variables and function names are case sensitive. This means that capitalization matters. `MYVAR` is not the same as `MyVar` or `myvar`. It is possible to have multiple distinct variables with the same name but different casing.

**Things to remember:**

1. Variable names can be made up of letters, numbers and the underscore character ( _ ).
2. They can't start with a number. You can use anything from single letters (such as a) to long sentences for variable names.
3. A variable can't contain a space, so use an underscore to separate words.
4. It is not allowed to use keywords as variable name. Keywords are some reserved words for C/C++ like cout, for, if etc.

**Examples**:

```cpp
int someVariable = 10;
int Total_price = 100;
```

**Question**: A car's MPG can be calculated using the following formula:

MPG = miles driven / gallons of gas used

For a given gallon value **G**, write a program that assigns the value 100 to the total number of miles he or she has driven and **G** to the gallons of gas used , then the program should calculate and display the car's MPG.

```cpp
#include <iostream>
using namespace std;
int main()
{
  int miles_driven=100;
  int gallon_of_gas_used;
  cin>>gallon_of_gas_used;
  int MPG=miles_driven / gallon_of_gas_used;
  cout<<MPG;
  //write your code here
}
```

Write a program that adds 100 and '**a**' (where a is keyboard input) and prints the result.

```
#include <iostream>
using namespace std;
int main()
{
  int a;
  cin>>a;
  cout<<a+100;
  //write your code here
}
```

Find the result of the following expressions and display the results in separate lines.

1. 80 % 16
2. 23 + 15 – 20
3. 4 * 4 * a ,where '**a**' is a keyboard input
4. $15^2$ % 6
5. $25^3$

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

int main()
{
  int a;
  cin>>a;
  cout<<80%16<<endl;
  cout<<23+15-20 <<endl;
  cout<<4*4*a<<endl;
  cout<< (15 * 15) % 6<<endl;
  cout<<25*25*25;
  //write your code here
}
```

Write a program to calculate and display the average of numbers 50 and '**a**', where a is a keyboard input.

```
#include <iostream>
using namespace std;
int main()
{
  int a;
  cin>>a;
  cout<<(a+50)/2;
  //write your code here
}
```

Convert the following instructions into a single line equation and print each result in a separate line.
1. Add 15 and 30, then multiply the result by 5, then subtract 10 from the result and divide it by 5.
2. Divide 10 by 5, then multiply the result by 3, then add **'a'** (where **'a'** is a keyboard input) to the result.
3. Add 45 and 15, then multiply the result by 2, then subtract 100 from the result

```
#include <iostream>
using namespace std;
int main()
{
  int a;
  cin>>a;
  int b=(((15+30)*5)-10)/5;
  int c=((10/5)*3)+a;
  int d=((45+15)*2) -100;
    cout<<b <<endl;
    cout<< c<<endl;
    cout<< d<<endl;
  //write your code here
}
```

Write a program to calculate and display the area of a square of side where side is a keyboard input.

```
#include <iostream>
using namespace std;
int main()
{
  int side;
  cin>>side;
  int area=side*side;
  cout<<area;
  //write your code here
}
```

Consider a rectangle of length **L** cm (where **L** is a keyboard input) and width 5 cm. Write a program to find and display the area of this rectangle using variables.

```
#include <iostream>
using namespace std;
int main()
{
  int length,width=5;
  cin>>length;
  cout<<length*width;
  //write your code here
}
```

Write a program to print the sum of squares of x and y, where x = 6 and y is a keyboard input .

```
#include <iostream>

#include<math.h>

using namespace std;

int main()

{

int x=6;

int y;

cin>>y;

cout<<(pow(6,2)+pow(y,2))<<endl;

return 0;

}
```

An object travelled 320 meters in **T** seconds (where **T** is a keyboard input), write a program to display the speed of the object.

```
#include <iostream>
using namespace std;
int main()
{
  int dist=320,time;
  cin>>time;
  int speed=dist/time;
  cout<<speed;
  //write your code here
}
```

Write a program that calculates and prints the value of Q according to the given formula:
Q = Square of Z, where Z = [(2 * C * D)/H]
Fixed values of C and D are 50 and 30 respectively and H is a keyboard input.

```
#include <iostream>
using namespace std;
int main()
{
  int H;
  cin>>H;
  int C=50,D=30;
  int Z=(2*C*D)/H;
  int Q=Z*Z;
  cout<<Q;
  //write your code here
}
```

Write a program that calculates simple interest that applies to the amount of Rs.10000 over a period of **'T'** years with interest rate 6%. Only the integer part of the simple interest should be displayed.
Simple interest = (Principal * Rate * Time) / 100 ,where Time **'T'** is a keyboard input

```
#include <iostream>
using namespace std;
int main()
{
  int P=10000,R=6;
  int T;
  cin>>T;
  int interest=(P*R*T)/100;
  cout<<interest;
  //write your code here
}
```

**Strings**

In C/C++, strings are collection of characters. Strings can have any keyboard character in them and can be as long as you want. String values can be used just like integer or float values.
A string can be represented by entering text between two double quotation marks. You can store strings in variables in C++ as below.

```
string new_string = "C++ is fun !!"
```

The quotes tell C/C++ ,where the string begins and ends. They are not part of the string value's text.
Note:
In C, strings are defined as an array of characters. Declaring a string is as simple as declaring a one dimensional array. Below is the basic syntax for declaring a string.
char str_name[size];
char str[] = "myPerfectice";
To read an input string: scanf("%s",str);
To print a string: printf("%s",str);
**Question**: Create a variable new_string (where new_string is a keyboard input), which stores the string value and then display the string value.
For example, input string new_string="Morning!" and output is "Morning!".

```
#include <iostream>
using namespace std;
int main()
{
  string new_string;
  cin>>new_string;
  cout<<new_string;
  //write your code here
}
```

**Multiline Strings**

For example, multiline_string = "C++ is a high-level
programming language"
A way to create a multiline string is to use '\n' which indicates a new line.
For example: multiline_string = "'C++ is a high-level \nprogramming language"'

**Question**: Write a program to print below string.
Salesman: Good evening, sir. How can I help you?
Customer: I have come to buy a pair of shoes.

```
#include <iostream>
using namespace std;
int main()
{
  string s="Salesman: Good evening, sir. How can I help you?\nCustomer:
I have come to buy a pair of shoes.";
  cout<<s;
  //write your code here
}
```

**Escape (ignore) Character**

If the string itself contains a double quote, C++ compiler may get confused about the quotes inside the string. So use escape character to display the additional quote.
For example, to print "He said,"I am fine"" we should write printf("He said,\"I am fine\"")
**Question**: Write a program to print "He said,"it's over, isn't it?""

```
#include <iostream>
using namespace std;
int main()
{
  cout<<"He said,\"it's over, isn't it?\"";  //write your code here
}
```

**String Concatenation**

You can combine string values with operators to make expressions, just as you did with integer and float values. When you combine two strings with the + operator, it's called string concatenation.
The + operator works differently on string and integer values because they are different data types. All values have a data type. The data type of the value "Hello" is string. The data type of the value 5 is an integer. The data type tells C++ what operators should do when evaluating expressions. The + operator concatenates string values, but adds integer values.
For Example:
s1="String 1";
s2="String 2"
result = s1 + s2;
In the C Programming Language, the **strcat function** appends a copy of the string pointed to by *s2* to the end of the string pointed to by *s1*. It returns a pointer to *s1* where the resulting concatenated string resides.(To use strcat() function, we need to include the header file #include <string.h>)
For Example:
char example[100] = "my";
printf("%s", strcat(example, "Perfectice"));
Above program will print myPerfectice.
**Question**: Print the single string by joining the two words as word1="Good " and word2, where word2 is a keyboard input .

For example, if input word w2="evening", so output would be Good evening.

```
#include <iostream>
using namespace std;
int main()
{
  string w1="Good";
  string w2;
  cin>>w2;
  cout<<w1+" "+w2;
  //write your code here
}
```

**Comments**

Comments are portions of the code ignored by the compiler which allow the user to make simple notes in the relevant areas of the source code. Comments come either in block form or as single lines.

Single-line comments (informally, C++ style), start with // and continue until the end of the line. If the last character in a comment line is a \ the comment will continue in the next line.

Multi-line comments (informally, C style), start with /* and end with */.

```
//This line is commented.
/*
this block
is commented
*/
```

**Question**: Write a program to print the result of the mathematical operation 36 + **'a'** ,where **'a'** is a keyboard input. Add a comment on top to describe the program.

```
#include <iostream>
using namespace std;
int main()
{
  int a;
  cin>>a;
  cout<<a+36;
  //write your code here
}
```

**Functions**

A function is a group of statements that together perform a task. Every C++ program has at least one function, which is **main()**, and programs can define additional functions.

You can divide up your code into separate functions. How you divide up your code among different functions is up to you, but logically the division usually is such that each function performs a specific task.

A function **declaration** tells the compiler about a function's name, return type, and parameters. A function **definition** provides the actual body of the function.

```
// function returning the max between two numbers
int max(int num1, int num2) {
   // local variable declaration
   int result;
```

```
   if (num1 > num2)
      result = num1;
   else
      result = num2;
   return result;
}
```

**Question**: Write a program to read two strings and print the concatenated string of two input words.

```
#include <iostream>
using namespace std;
int join(string a,string b)
{
  cout<<a+b;
  //write your code here
}
int main()
{
  string a,b;
  cin>>a>>b;
  join(a,b);
  return 0;
}
```

**Array**

C++ provides a data structure, the array, which stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data but it is often more useful to think of an array as a collection of variables of the same type.

Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

**Question**: Create an array animals containing the below elements and print the **n** array elements in separate lines where **n** is a keyboard input.

Elements are Tiger, Lion, Monkey, Elephant and Deer.

```
#include <iostream>
#include<string>
using namespace std;
int main()
{
  string arr[]={"Tiger","Lion","Monkey","Elephant","Deer"};
  int n;
  cin>>n;
  for(int i=0;i<n;i++){
    cout<<arr[i]<<endl;
  }
  //write your code here
}
```

In C programming, you can create array of an array known as multidimensional array.

**For example,**

```
int x[3][4];
```

Here, x is a two-dimensional (2D) array. The array can hold 12 elements. You can think the array as table with 3 rows and each row has 4 column.

Similarly, you can declare a three-dimensional (3d) array. For example,

```
float y[2][4][3];
```

Here,The array y can hold 24 elements.

You can think this example as: Each 2 elements have 4 elements, which makes 8 elements and each 8 elements have 3 elements. Hence, the total number of elements is 24.

int c[2][3] = {{1, 3, 0}, {-1, 5, 9}};
int c[][3] = {{1, 3, 0}, {-1, 5, 9}};
int c[2][3] = {1, 3, 0, -1, 5, 9};

Above code are three different ways to initialize a two dimensional arrays.

**Question**:

Create an array to store the data given in the following table and print the array elements as given in the expected output.

| | | |
|---|---|---|
| 14 | N | 98 |
| 36 | 78 | 25 |
| 25 | 66 | 78 |
| 78 | 34 | 81 |

Note: N is a keyboard input

```
#include <iostream>
using namespace std;
int main ()
{
  int data[4][3]={{14,-1,98},{36,78,25},{25,66,78},{78,34,81}};
  int N;
  cin>>N;
  data[0][1]=N;
  for(int i=0;i<4;i++){
    for(int j=0;j<3;j++){
      if(data[i][j]==-1){
        cout<<N<<" ";
      }else{
        cout<<data[i][j]<<" ";
      }
    }
    cout<<endl;
  }
  //write your code here
}
```

**Enumeration (or enum)**

Enumeration (or enum) is a user defined data type in C. It is mainly used to assign names to integral constants, the names make a program easy to read and maintain. The keyword 'enum' is used to declare new enumeration types in C and C++. Following is an example of enum declaration.

enum season { spring, summer, autumn, winter };
Here, the name of the enumeration is `season`.
And `spring`, `summer` and `winter` are values of type `season`.
By default spring is 0, summer is 1 and so on. You can change the default value of an enum element during declaration (if necessary).

```
enum season
{  spring = 0,
   summer = 4,
   autumn = 8,
   winter = 12
};
```

When you create an enumerated type, only blueprint for the variable is created. Here's how you can create variables of enum type.

```
enum boolean { false, true };
// inside function
enum boolean check;
```

Here, a variable check of type enum boolean is created.

**Question:**

1. Define week as enum datatype and declare weekdays in it.
2. Initialize a variable `today` as enum type, assign value "Wednesday".
3. Print the value of `today + n; where n is a keyboard input as 1 or 2 or 3`

```
#include <iostream>
enum week { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday };
using namespace std;
int main() {
    week today = Wednesday;
    int n;
    cin >> n;

    week tomorrow = static_cast<week>((today + n) % 7);
    cout <<tomorrow <<endl;

    return 0;
}
```

**Structure**

Structure is a collection of variables of different data types under a single name.

**For example:** You want to store some information about a person: his/her name, citizenship number and salary. You can easily create different variables name, citNo, salary to store these information separately.

However, in the future, you would want to store information about multiple persons. Now, you'd need to create different variables for each information per person: name1, citNo1, salary1, name2, citNo2, salary2

The struct keyword defines a structure type followed by an identifier (name of the structure). Then inside the curly braces, you can declare one or more members (declare variables inside curly braces) of that structure. For example:

```
struct Person
{
    char name[50];
    int age;
    float salary;
```

```
};
```
Here a structure `person` is defined which has three members: `name`, `age` and `salary`.
When a structure is created, no memory is allocated.
Once you declare a structure `person` as above. You can define a structure variable as:
```
Person bill;
```
Here, a structure variable `bill` is defined which is of type structure `Person`.
When structure variable is defined, only then the required memory is allocated by the compiler.
The members of structure variable is accessed using a dot (.) operator.
Suppose, you want to access `age` of structure variable `bill` and assign 50 to it. You can perform this task by using the code below:
```
bill.age = 50;
```
**Union**
Union follow the same syntax as structures. However there is a major distinction between them in terms of storage. In structure, each member has its own storage location, whereas all the members of a union use the same location.
**Question:**
Define a structure **struct personal** that would contain person's first name, date of joining and salary. Using this structure, write a program to read this information for one person and print the same in the format given in the expected output.

```
#include<iostream>
using namespace std;
struct Date {
    int day;
    char month[20];
    int year;
};
struct personal {
    char firstName[50];
    Date dateOfJoining;
    float salary;
};
int main() {
    personal person;
    cin.getline(person.firstName, 50);
    cin >> person.dateOfJoining.day >> person.dateOfJoining.month >>
person.dateOfJoining.year;
    cin >> person.salary;
    cout << person.firstName << " ";
    cout <<  person.dateOfJoining.day << " " << person.dateOfJoining.month << " "
<< person.dateOfJoining.year << " ";
    cout << person.salary << endl;

    return 0;
}
```

---

Write a program that defines two strings **str1** and **str2** separately (where **str1** and **str2** is a keyboard input) and then prints the both strings as a single string.
For example, if string1="I don't " and string2="know",output would be as "I don't know"

```
#include <iostream>
```

```cpp
#include <string>
using namespace std;

int main() {
    string str1, str2;
    getline(cin,str1);
    getline(cin,str2);

    string comb = str1 + str2;

    cout << comb<< endl;

    return 0;
}
```

Write a program to print the following :-
Alice: Hurrah! Only ten days to the holidays.
Bob: I know. I've been counting the days.

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

int main() {
    cout << "Alice: Hurrah! Only ten days to the holidays.\n";
    cout << "Bob: I know. I've been counting the days.\n";

    return 0;
}
```

Write a program to take 3 integers as input and find average of three numbers and then print only the integer part of the average.

**Sample input:**

15
23
25

**Expected output:**

Average is 21

```cpp
#include <iostream>
using namespace std;
int main()
{
  int a,b,c;
  cin>>a>>b>>c;
  int avg=(a+b+c)/3;
  cout<<"Average is "<<avg;
  //write your code here
}
```

Write a program that reads first name, lastname and age of a person and prints his/her details.

**Sample input:**
Albert
Einstein
30
**Expected Output:**
Albert Einstein is 30 years old

```
#include <iostream>
using namespace std;
int main()
{
  string first,last;
  int age;
  cin>>first;
  cin>>last;
  cin>>age;
  cout<<first+" "+last << " is "<<age<<" years old";
  //write your code here
}
```

Create an array `fruits` having items Orange, Apple, Mango, Pineapple, Strawberry and Banana.
1. Print the third item in the array of fruits.
2. Print the '**n**' item in the array of fruits ,where n is a keyboard input

```
#include <iostream>
using namespace std;
int main()
{
  string fruits[] =
{"Orange","Apple","Mango","Pineapple","Strawberry","Banana"};
  cout<<fruits[2]<<endl;
  int n;
  cin>>n;
  cout<<fruits[n-1]<<endl;
  //write your code here
}
```

Given an array num = {6, 7, 3, 8, 1, 5, 16, 13, 11, 19, 2, 4, 12, 9, 17, 10}
1. Print 8th and 12th element of the array separated by a space.
2. Replace the second element of array with **'a'**, where **'a'** is a keyboard input.
3. Print the sum of second, third and sixth elements of the array in a separate line.

```
#include <iostream>
using namespace std;
int main()
{
  int num[] = {6, 7, 3, 8, 1, 5, 16, 13, 11, 19, 2, 4, 12, 9, 17, 10};
  cout<<num[7]<<" "<<num[11]<<endl;
  int a;
  cin>>a;
  num[1]=a;
```

```
  cout<<num[1]+num[2]+num[5]<<endl;
  //write your code here

}
```

For a given table of 2D array,

| | | |
|---|---|---|
| 56 | 85 | 36 |
| 96 | 37 | 82 |
| 46 | 34 | 95 |
| 63 | 84 | 52 |

write a program to print the elements at the following positions in separate lines.
1. First element of the third row
2. Third element of the second row
3. First element of the first row

```
#include <iostream>
using namespace std;
int main()
{
  int tab[4][3] = { 56, 85, 36, 96, 37, 82, 46, 34, 95, 63, 84, 52 };
cout<<tab[2][0]<<"\n"<<tab[1][2]<<"\n"<<tab[0][0]<<endl;
  //write your code here

}
```

Write a program to store the roll no, name and age of a student. Make roll no = 1 and take user input for name and age and then print the details of the student using structure.
Details should be displayed in the format provided in expected output.

```
#include <iostream>
#include <string>
using namespace std;
struct Student {
    int rollNo;
    string name;
    int age;
};

int main() {
    Student student;
    student.rollNo = 1;
    getline(cin, student.name);
    cin >> student.age;
    cout << "Roll no: " << student.rollNo << endl;
    cout << "Name: " << student.name << endl;
    cout << "Age: " << student.age << endl;
  //write your code here
}
```

## IF Statements

The program execution starts at the top instruction in the program and moves straight down, executing each instruction in order. But with some statements, you can control the flow of the program based on conditions. These kinds of statements are flow control statements, since they change the flow of the program execution as it moves around your program. Flow control statements include conditional statements and loops.

In day to day life, we make decisions based on our situation and each of our decision will have a certain effect on what you end up doing. Every decision we make can be modeled as a series of true and false statements.

In programming also, we often ask yes or no questions, and decide to do something based on the answer. These sorts of questions are called **conditions**.

**Conditional statements** are the digital equivalent of the decisions we make, where our code does something different depending on whether something is **true** or **false**.

For example, in a situation where we need to do something only if some conditions are True, we can use if statements.

Structure of if statement is the word if followed by a condition and colon. On next line, write commands to be run.

```
if(marks == 50){
      cout<<("Good"); }
```

if statement will run if its condition evaluates to True. If the condition is False, the code in the if block is skipped. Using if statements, you can make the program run certain code only when you want it to.

**Question**: Write a program that reads an integer from user and prints a message "Invalid" if the input value is less than zero, otherwise print "Valid".

```
#include <iostream>
using namespace std;
int main()
{
  int num;
  cin>>num;
  if(num<0){
    cout<<"Invalid";
  }else{
    cout<<"Valid";
  }
  //write your code here

}
```

## Block of statements

A **block** of statements is a group of statements that is treated by the compiler as if it were a single statement. Blocks begin with a { symbol and end with a } symbol and the statements to be executed are placed in between. Blocks can be used at any place where a single statement is allowed. No semicolon is needed at the end of a block.

```
if(number == 0){
    printf("Not valid");
    printf("Number must be greater than zero");
    printf("Try another");
```

```
}
```

**Question**: Write a program that reads a number entered by the user ( which is 4 or 5) and prints previous number of it in separate lines. If number is not equal to user input, then don't print anything.
For example, if input number is 5 then output is 1, 2, 3 and 4 in separate lines. If number is not equal to 5, then don't print anything.

```
#include <iostream>
using namespace std;
int main()
{
  int num;
  cin>>num;
        if(num ==4 || num ==5){
      for(int i=1;i<num;i++){
        cout<<i<<endl;
      }
    }
  //write your code here

}
```

---

**Comparison Operators**

The Boolean data type has only two values: true or false.
Boolean values can be stored in variables just like the other data types:

```
bool isValid = true;
```

We use symbols to create our conditions such as equal to, greater than, and less than etc.
When we use comparison operator, the program compares things and tells us whether the criteria set by the comparison are True (yes) or False (no).

```
age > 15
```

The condition `age > 15` asks, "Is the value stored in `age` greater than 15?" If so, then the condition evaluates to true. If not, the condition evaluates to false.

**Comparison operators are:**

| Operator | Operation |
|---|---|
| < | Less than |
| <= | Less than or equal to |
| > | Greater than |
| >= | Greater than or equal to |
| == | Equal to |
| != | Not equal to |

Be careful not to confuse the assignment operator(=) and the equal to comparison operator(==). The equal sign(=) is used in assignment statements to store a value to a variable, whereas the double equal sign(==) is used in expressions to see whether two values are equal. It's easy to accidentally use one when you mean to use the other.
It might help to remember that both the equal to comparison operator(==) and the not equal to comparison operator(!=) have two characters.

**Question**: Write a program that reads an integer value and prints
1. "Positive" if the value is greater than 0
2. "Negative" if the value is less than 0
3. "Zero" if the value is equal to 0

```
#include <iostream>
```

```
using namespace std;
int main()
{
  int user_input;
  cin>>user_input;
  if(user_input > 0){
    cout<<"Positive";
  }else if(user_input<0){
    cout<<"Negative";
  }else{
    cout<<"Zero";
  }
  //write your code here

}
```

**IF-ELSE statements**

We can also use if statements to do something when a condition is not true. You can think of "if-else" statements as saying, "If something is true, then do this; or else, do something different."
For example:

```
if(limit >= 10)
    cout<<("You have reached your limit");
else
    cout<<("Go to next");
```

If the limit variable is greater than or equal to the value 10, then the if statement's condition is true and if block tells the user that they have reached their limit. However, if this condition is false, then the compiler executes the else block and the string "Go to next" is printed to the screen.

**Question**: Write a program to print the result of a student based on their total marks. The user inputs the total marks of a student. If the total marks is greater than 120, then the program should display "You are qualified", otherwise "Not qualified" without quotes.

```
#include <iostream>
using namespace std;
int main()
{
  int marks;
  cin>>marks;
  if(marks>120){
    cout<<"You are qualified";
  }else{
    cout<<"Not qualified";
  }
  //write your code here

}
```

**Combining Conditions**

You can combine two or more conditions by using the keywords **and** and **or** which produces shorter and simpler code.
Suppose if you are given a number from 1 to 20 and you need to write a program that checks if that number is a multiple of 5 or not. We know that 5, 10, 15 and 20 are multiples of 5. We don't need to check condition for these numbers separately, we can write the code as

```
if(number == 5 || number == 10 || number == 15 || number == 20)
    cout<<("Multiple of 5");
else
    cout<<("Not a multiple of 5");
```

The result of two operations combined by ||(OR) will be true, if any of the condition is satisfied.
The result of two operations combined by &&(AND) will be true, only if both conditions are satisfied.
True && True = True
True && False = False
False && False = False
True || True = True
True || False = True
False || False = False
**Question**: Write a program that prints "It's a holiday" if the day given by the user is Sunday or Saturday, otherwise it prints "Not a holiday".

```
#include <iostream>
using namespace std;
int main()
{
  string day;
  cin>>day;
  if(day=="Sunday"||day=="Saturday"){
    cout<<"It's a holiday";
  }else{
    cout<<"Not a holiday";
  }
  //write your code here

}
```

**<u>else if Statements</u>**
Now we can expand our if-else statement further with else if statements. You can think of else if statements as saying, “If this is true, do this. Or else if this next condition is true, do that. Or else if none of them is true, do this last thing.” Instead of making two decisions, we can now compare more than two situations by using else if  in between if and else statements.
For example:

```
if(number > 0)
    cout<<("Positive");
else if(number == 0)
    cout<<("Zero");
else
    cout<<("Negative");
```

In this example, if the number variable is greater than 0, then the if statement’s condition is true and the if block tells the user that the number is positive. However, if this condition is false, then compiler tries the else if statement’s condition next. If number is equal to 0, then the string 'Zero' is printed to the screen. If both are false, then the code tells the user that the number is negative.

You can have as many else if statements as you want. You can also leave off the else block if you don't need one and just have if-else if statements.
Question: Write a program that takes the marks of a student as input and displays a message based on the following data:-

| Total marks | Message |
|---|---|
| Greater than 80 | Outstanding |
| Between 61-80 | Excellent |
| Between 41-60 | Good |
| Less than or equal to 40 | Not qualified |

```
#include <iostream>
using namespace std;
int main()
{
  int marks;
  cin>>marks;
  if(marks>80){
    cout<<"Outstanding";
  }else if(marks>60 && marks<80){
    cout<<"Excellent";
  }else if(marks>40 && marks<60){
    cout<<"Good";
  }else if(marks <=40){
    cout<<"Not qualified";
  }
  //write your code here

}
```

**The switch statement**

The if..else..if ladder allows you to execute a block of code among many alternatives. If you are checking on the value of a single variable in if...else...if, it is better to use a switch statement. The switch statement is often faster than nested if...else (not always). Also, the syntax of the switch statement is cleaner and easy to understand.
Syntax of the switch statement is as follows:

```
switch (n)
{
    case constant1:
        // code to be executed if n is equal to constant1;
        break;
    case constant2:
        // code to be executed if n is equal to constant2;
        break;
        .
        .
        .
    default:
```

```
        // code to be executed if n doesn't match any constant
}
```

When a case constant is found that matches the switch expression, control of the program passes to the block of code associated with that case.

Suppose, the value of *n* is equal to *constant2*. The compiler executes the statements after case constant2 : until break is encountered. When a break statement is encountered, switch statement terminates.

**Question**:

Write a program that takes an input from user(user enters 1 for addition and 2 for subtraction). Then the user enters two numbers. You have to then output the result (number1 + number2) or (number1 - number2) based on the user input.

```
#include <iostream>
using namespace std;
int main()
{
  int op;
  int firstNumber, secondNumber;
  cin >> op;
  cin >> firstNumber >> secondNumber;
  switch(op){
    case 1:
      cout<<firstNumber+secondNumber;
      break;
    case 2:
      cout<<firstNumber-secondNumber;
      break;
    default:
      cout<<"error";
  }
  //write your code here

}
```

Write a program that reads a number from user and prints "You have reached the maximum limit" if the number is greater than 60 else print the number itself.

```
#include <iostream>
using namespace std;
int main()
{
  int num;
  cin>>num;
  if(num>60){
    cout<<"You have reached the maximum limit"<<endl;
  }else{
    cout<<num;
  }
  //write your code here

}
```

Write a program that reads an amount of money as input from the user and if the amount is greater than 1000, the program should print "You are rich", otherwise "You are not rich" should be printed.

```
#include <iostream>
using namespace std;
int main()
{
  int num;
  cin>>num;
  if(num>1000){
    cout<< "You are rich";
  }else{
    cout<< "You are not rich";
  }
  //write your code here

}
```

Write a program that takes two numbers as input and displays the greatest of two numbers without using else option.

```
#include <iostream>
using namespace std;
int main()
{
  int x,y;
  cin>>x>>y;
  (x > y) ? cout<<x : cout<<y;
  //write your code here

}
```

Write a program that takes two numbers as input and displays the greatest of two numbers using else option.

```
#include <iostream>
using namespace std;
int main()
{
  int x,y;
  cin>>x>>y;
  //(x > y) ? cout<<x : cout<<y;
  if(x>y){
    cout<<x;
  }
  if(y>x){
    cout<<y;
  }
```

```
  //write your code here

}
```

Write a program that takes a number from user as input. If the number is less than 10, it should print "Too few" and if it is greater than 70, it should print "Too many".

```
#include <iostream>
using namespace std;
int main()
{
  int user_input;
  cin>>user_input;
  if(user_input<10){
    cout<<"Too few";
  }else if(user_input>70){
    cout<<"Too many";
  }
  //write your code here

}
```

Write a program that takes an input from the user. If the input is between 100 and 500 or between 1000 and 2000 (both inclusive), print "Yes", otherwise print "No".

```
#include <iostream>
using namespace std;
int main()
{
  int money;
  cin>>money;
  if(money>=100 && money <= 500 || money >= 1000 && money<=2000){
    cout<<"Yes";
  }else{
    cout<<"No";
  }
  //write your code here

}
```

Write a program that reads marks of a student as input and prints "Above average" if the input is between 50 and 60 (both inclusive), otherwise doesn't print anything.

```
#include <iostream>
using namespace std;
int main()
{
  int marks;
  if(marks >= 50 && marks <=60){
```

```
    //cout<<"Above average";
  }
  else{
    cout<<"Above average";
  }
  //write your code here
}
```

Write a program that reads an integer value and prints "Out of range" if the value is less than zero or greater than 100, otherwise it prints "Proceed".

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

int main() {
  int val;
  cin >> val;

  if (val < 0 || val > 100) {
    cout << "Out of range";
  } else {
    cout << "Proceed";
  }
  return 0;
}
```

Admission to a high school course is subjected to the following conditions:-
1. Marks in mathematics greater than or equal to 60 and
2. Marks in science greater than or equal to 50 and
3. Marks in language is greater than or equal to 40
   OR simply
Total in Mathematics and Science is greater than or equal to 150.
Write a program that takes 3 inputs. The inputs are mathematics, science and language marks respectively. You have to output "Eligible" if the student is eligible, otherwise output "Not Eligible".

```
#include <iostream>
using namespace std;
int main()
{
  int mathematics,science,language;
  cin>>mathematics>>science>>language;
 if(mathematics>=60&&science>=50&&language>=40) {
    cout<<"Eligible";
  }else if((mathematics + science)>=150) {
    cout<<"Eligible";
  }else{
    cout<<"Not Eligible";
 }
  //write your code here
}
```

Write a program to check if a character is a vowel or consonant. Entered character should be in lower case.
Output "Vowel" or "Consonant" based on the character entered.

```
#include <iostream>
using namespace std;
int main()
{
  char character;
  cin>>character;

if(character=='a'||character=='e'||character=='i'||character=='o'||character=='u'
){
    cout<<"Vowel";
  }else{
    cout<<"Consonant";
  }
  //write your code here

}
```

Write a program to print the number of days in a month if a month number is given.
Input: For January, give input as integer 1, for February 2 and so on.
You have to output "31 Days" for months having 31 days, "30 Days" for months having 30 days and "28 or 29 Days" for February

```
#include <iostream>
using namespace std;
int main()
{
  int month;
  cin>>month;
  if(month==1||month==3||month==5||month==7||month==8||month==10||month==12){
    cout<<"31 Days";
  }else if(month==2){
    cout<<"28 or 29 Days";
  }else{
    cout<<"30 Days";
  }
  //write your code here

}
```

Write a program that begins by reading the denomination of a banknote from the user. Then your program should print the name of the individual that appears on the banknote of the entered amount. An appropriate error message("Invalid Entry") should be displayed if no such note with entered denomination exists.

| **Individual** | **Amount** |
| --- | --- |
| George Washington | $1 |

Thomas Jefferson $2
Abraham Lincoln $5
Alexander Hamilton $10
Andrew Jackson $20
Ulysses S. Grant $50
Benjamin Franklin $100

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

int main() {
    int denom;
    cin >> denom;

    string Name;

    switch (denom) {
        case 1:
            Name = "George Washington";
            break;
        case 2:
            Name = "Thomas Jefferson";
            break;
        case 5:
            Name = "Abraham Lincoln";
            break;
        case 10:
            Name = "Alexander Hamilton";
            break;
        case 20:
            Name = "Andrew Jackson";
            break;
        case 50:
            Name = "Ulysses S. Grant";
            break;
        case 100:
            Name = "Benjamin Franklin";
            break;
        default:
            Name = "Invalid Entry";
    }

    cout <<Name << endl;
  //write your code here

}
```

Create a program that reads a month and day from the user. The user will enter the name of the month as a string(First character uppercase, rest lowercase) followed by the day as an integer. Then your program should display the season associated with the date that was entered.

| Season | First day |
|---|---|
| Spring | March 20 |
| Summer | June 21 |
| Fall | September 22 |
| Winter | December 21 |

Your program should display "Winter Season", "Summer Season", "Fall Season" or "Spring Season" based on the input

```
#include <iostream>
using namespace std;
int main()
{
  string month;
  int date;
  cin>>month;
  cin>>date;
 if ((month == "December" && date >= 21) || (month == "January") || (month == "February") || (month == "March" && date < 20)) {
        cout << "Winter Season";
    } else if ((month == "March" && date >= 20) || (month == "April") || (month == "May") || (month == "June" && date < 21)) {
        cout << "Spring Season";
    } else if ((month == "June" && date >= 21) || (month == "July") || (month == "August") || (month == "September" && date < 22)) {
        cout << "Summer Season";
    } else if ((month == "September" && date >= 22) || (month == "October") || (month == "November") || (month == "December" && date < 21)) {
        cout << "Fall Season";
    } else {
        cout << "Invalid";
 }
  //write your code here

}
```

Write a program that takes a number(1 to 7) as input from the user and prints its corresponding week day.
You have to output "Sunday" for 1, "Monday" for 2 and so on.

```
#include <iostream>
using namespace std;
int main()
{
  int num;
  cin>>num;
switch(num){
  case 1:
```

```
    cout<<"Sunday";
    break;
   case 2:
    cout<<"Monday";
    break;
   case 3:
    cout<<"Tuesday";
    break;
   case 4:
    cout<<"Wednesday";
    break;
   case 5:
    cout<<"Thursday";
    break;
   case 6:
    cout<<"Friday";
    break;
   case 7:
    cout<<"Saturday";
    break;
  default:
    cout<<"Invalid";
}
  //write your code here

}
```

Write a program that takes values of length and breadth of a rectangle from the user and check if this rectangle is a square or not.
You have to output "Yes, it is a square" or "No, it is a rectangle" based on the input.

```
#include <iostream>
using namespace std;
int main()
{
  int length,breadth;
  cin>>length>>breadth;
  if(length==breadth){
    cout<<"Yes, it is a square";
  }else{
    cout<<"No, it is a rectangle";
  }
  //write your code here

}
```

Write a program that takes 3 integers as input and then display the smallest number among them.

```
#include <iostream>
using namespace std;
int main()
{
  int first,second,third;
  cin>>first>>second>>third;
  if(first< second && first<third){
    cout<<first;
  }else if(second<first && second<third){
    cout<<second;
  }else{
    cout<<third;
  }
  //write your code here

}
```

You are given an array containing the elements 5, 6, 2, 1, 4, 3. You have to print the square of all these elements separated by spaces.

```
#include <iostream>
using namespace std;
int main()
{
  int num[] = {5, 6, 2, 1, 4, 3};
   for(int i=0;i<6;i++){
    cout<<num[i]*num[i]<<" ";
   }
  //write your code here

}
```

Print the multiplication table of a given number using for loop.
For example if you are given the number 2, you will have to print:-
1 x 2 = 2
2 x 2 = 4
.
.
.
10 x 2 = 20

```
#include <iostream>
using namespace std;
int main()
{
  int num;
  cin>>num;
  for (int i = 1; i <= 10; ++i) {
```

```
        cout << i << " X " << num << " = " << (i * num) << endl;
  }
  //write your code here

}
```

Write a program to print all multiples of 10 in the range of 1 and 100(including 100) in separate lines

```
#include <iostream>
using namespace std;
int main() {
    for (int i = 10; i <= 100; i += 10) {
        cout << i << endl;
    }
  //write your code here
}
```

Write a program to print the number series 2, 4, 8,..... ,1024. Each number should be displayed in a separate line.

```
#include <iostream>
using namespace std;
int main()
{
  int n=2;
  while(n<=1024){
    cout << n <<endl;
    n*=2;
  }
  //write your code here
}
```

Write a program to print the number series 0, 5, 10, 15, 30, 35, 70, 75, 150, 155, 310, 315, 630, 635. Each number should be displayed in a separate line.

```
#include <iostream>
using namespace std;
int main() {
    int series[] = {0, 5, 10, 15, 30, 35, 70, 75, 150, 155, 310, 315, 630,
635};
    int length = sizeof(series) / sizeof(series[0]);

    for (int i = 0; i < length; ++i) {
        cout << series[i] <<endl;
    }

    return 0;
}
```

Create an array of month names in order and print each month with it's sequence number.
For example, your output should look like:-
1 January
2 February
.....
12 December

```
#include <iostream>
using namespace std;
int main()
{
  string
month[]={"January","February","March","April","May","June","July","August
","September","October","November","December"};
  for(int i=0;i<12;i++){
    cout<<i+1<<" "<<month[i]<<endl;
  }
  //write your code here
}
```

Take 10 integers as input from user using loop and print their average value on the screen.
The average value should be displayed correct to one decimal place.

```
#include <iostream>
#include<iomanip>
using namespace std;
int main()
{
  int sum=0;
  int num;
  for(int i=0;i<10;i++){
    cin>>num;
    sum+=num;
  }
  double average = static_cast<double>(sum) / 10;
 cout<<fixed<<setprecision(1)<< average << endl;
  //write your code here
}
```

Take integer inputs from the user until the user presses q. Print the floor of their average (floor means truncate all the decimal part of number eg 5.2 should be printed as 5 only and 5.9 should also be 5 only ).

```
#include <iostream>
#include <string>

using namespace std;

int main() {
    int sum = 0;
    int count = 0;
```

```
    string input;
    while (true) {
        cin >> input;
        if (input == "q") {
            break;
        }
        sum += stoi(input);
        ++count;
    }

    if (count == 0) {
        cout << "error" << endl;
    } else {
        int average = sum / count;
        cout <<average << endl;
    }
}
```

Take 5 integers as input from user to make an array. Again take one input from the user and search it in the array
Delete that element, if found.
Output the modified array with each element in a different line.

```
#include <iostream>

using namespace std;

int main() {
    int arr[5];
    int numberToDelete;
    bool found = false;
    for (int i = 0; i < 5; ++i) {
        cin >> arr[i];
    }
    cin >> numberToDelete;
    for (int i = 0; i < 5; ++i) {
        if (arr[i] == numberToDelete) {
            found = true;
            for (int j = i; j < 4; ++j) {
                arr[j] = arr[j + 1];
            }
            break;
        }
    }
    for (int i = 0; i < (found ?4: 5); ++i) {
        cout << arr[i] << endl;
    }

    return 0;
}
```

Make two arrays one containing all even numbers and other containing all odd numbers in the range of 1 to 100 (including 1 and 100).
Output the complete even array separated by spaces, then in the next line output the odd array separated by spaces.

```
#include <iostream>
using namespace std;
int main() {
    int evenNumbers[100];
    int oddNumbers[100];
    int evenCount = 0;
    int oddCount = 0;
    for (int i = 1; i <= 100; ++i) {
        if (i % 2 == 0) {
            evenNumbers[evenCount++] = i;
        } else {
            oddNumbers[oddCount++] = i;
        }
    }
    for (int i = 0; i < evenCount; ++i) {
        cout << evenNumbers[i] << " ";
    }
  cout << endl;
    for (int i = 0; i < oddCount; ++i) {
        cout << oddNumbers[i] << " ";
    }
  //write your code here
}
```

Write a program that takes 2 inputs from the user. Output all the values starting from the first input and continuously incrementing by 2 upto the second input.
Output all the numbers in separate lines.

```
#include <iostream>
using namespace std;
int main()
{
  int start,end;
  cin>>start>>end;
  while(start<=end){
    cout<<start<<endl;
    start+=2;
  }
  //write your code here

}
```

**The for loop**

In programming, loops let you execute a portion of code over and over again until a particular task is complete. Think of the for statement as saying, "Execute the code in the following block a certain number of times."

To print numbers from 1 to 100, we don't need to write print command 100 times, instead we can create a single for loop which execute 100 times and in each execution it will print numbers from 1 to 100 one by one.

```
for(initializationStatement; testExpression; updateStatement) {
   // code
}
```

The initialization statement is executed only once at the beginning.

Then, the test expression is evaluated.

If the test expression is false, for loop is terminated. But if the test expression is true, codes inside body of for loop is executed and update expression is updated.

Again, the test expression is evaluated and this process repeats until the test expression is false.

Question: Write a program to print numbers from 20 to 30(Each number should be printed in a separate line)

```
#include <iostream>
using namespace std;
int main()
{
  for(int i=20;i<=30;i++){
    cout<<i<<endl;
  }
  //write your code here
}
```

**Looping array items**

We can use looping on array, by using index of the array. Assume you have a list of four friends - James, Monica, Adam and Rachel.

```
char friends_list[4][10] = {"James", "Monica", "Adam", "Rachel"};
```

In two dimesional array declaration, 4 indicates the total number of elements in the array and 10 indicates maximum characters in an element

To write a program to say 'Hello' to friends,

```
for(i=0;i<4;i++){
    cout<<"Hello "<<friends_list[i]<<endl; }
```

**Question**: Create an array of colors Red, Green, Blue, Black and Yellow and print all elements of colors in separate lines using for loop.

```
#include <iostream>
using namespace std;
int main()
{
  string colors[] = {"Red", "Green", "Blue", "Black", "Yellow"};
for(int i=0;i<5;i++){
  cout<<colors[i]<<endl;
}
  //write your code here

}
```

**The Break Statement**

A break statement tells the execution to jump immediately out of the for block to the first line after the end of the for block. The break statement is found only inside loops, such as in a for block.

For example, Suppose we have a random list of two digit numbers, if we need to check whether the number 44 is there in list, we can compare each number in the list with 44, if we find the number, no need to proceed further, So we can put a break statement.

```
for int i=0;i<size;i++){
   if (i == 44){
       cout<<("Found");
       break; } }
```

**Question**: We have a array of numbers {25, 33, 42, 28, 53, 66, 75, 36, 80, 40}, write a program to find sum of first five numbers in the array.

Output should be like "Sum of first five numbers is 181" without the quotes.

```
#include <iostream>
using namespace std;
int main()
{
  int arr[]={25, 33, 42, 28, 53, 66, 75, 36, 80, 40};
       int sum=0;
  for(int i=0;i<5;i++){
    sum+=arr[i];
  }
  cout<<"Sum of first five numbers is "<<sum;
  //write your code here

}
```

**Nested loops**

We can define loops inside another loop as:

```
for ( init; condition; increment ) {
   for ( init; condition; increment ) {
      statement(s);
   }
   statement(s); // you can put more statements.
}
```

**Question**: We have an array of numbers **num** = {16, 8, 12, 5, 7, 10, 11, 9, 13, 15}, write a program that asks user to input a number and if number is present in num, print numbers from 1 to the value of input in separate lines, otherwise print “Not found”

```
#include <iostream>
using namespace std;
int main(){
  int num[] = {16, 8, 12, 5, 7, 10, 11, 9, 13, 15};
  int user_input,f=0;
  cin>>user_input;
  for(int i=0;i<=10;i++){
    if(num[i]==user_input){
      f=1;
```

```
      for(int j=1;j<=user_input;j++){
        cout<<j<<endl;
      }
      break;
    }
  }
    if(f==0){
      cout<<"Not found";
    }
  return 0;
}
```

**The While Loop**

Unlike a for loop, which loops a specific number of times, a while loop repeats as long as a certain condition is True. When the execution reaches a while statement, it evaluates the condition next to the while keyword. If the condition evaluates to True, the execution moves inside the following block, called the while block. If the condition evaluates to False, the execution moves past the while block.

For example, program to increment the value of x and y based on certain conditions,

```
x = 28
y = 44
while (x < 35 and y < 50){
   x = x + 1;
   y = y + 1;
   cout<<x<<y;
}
```

**Question**: Write a program that takes two numbers x and y as input from the user and then print all the numbers between x and y(including x and y) using while loop. The numbers should be displayed in separate lines

```
#include <iostream>
using namespace std;
int main()
{
  int x,y;
  cin>>x>>y;
for(int i=x;i<=y;i++){
  cout<<i<<endl;
}
  //write your code here

}
```

**Functions**

In programming, function is a sequence of commands that can be reused together later in a program. Functions let you run the same code multiple times without having to copy and paste that code over and over again. Instead, you put that code inside a function and call the function whenever you need to. Because you write the code only once in the function, if the function's code has a mistake, you only have to change it in one place in the program.

The C++ standard library provides numerous built-in functions that your program can call. For example, function strcat() to concatenate two strings, function memcpy() to copy one memory location to another location and many more functions.
A C++ function definition consists of a function header and a function body. Various parts of function are as follows:-
**Return Type** – A function may return a value. The return_type is the data type of the value the function returns. Some functions perform the desired operations without returning a value. In this case, the return_type is the keyword void.
**Function Name** – This is the actual name of the function. The function name and the parameter list together constitute the function signature.
**Parameters** – A parameter is like a placeholder. When a function is invoked, you pass a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order and number of the parameters of a function. Parameters are optional; that is, a function may contain no parameters.
**Function Body** – The function body contains a collection of statements that define what the function does.

```
return_type function_name( parameter list ) {
   body of the function;
}
```

When you define a function, you specify the instructions for it to run in the following block. Unless you call the function, the instructions in the block will not execute. Later when you call a function, the code inside the block executes.

**Question**: Define a function to print the poem
Whose woods these are I think I know.
His house is in the village though;
He will not see me stopping here
To watch his woods fill up with snow.

```
#include <iostream>
using namespace std;
void print_poem()
{
  cout<<"Whose woods these are I think I know.\nHis house is in the village though;\nHe will not see me stopping here\nTo watch his woods fill up with snow.";
  //write your code here
}
int main()
{
  print_poem();
  return 0;
}
```

**Function Parameters**

Now let's look at function parameters. We can define a function to say Hello to a particular person by passing their name as function parameters.

```
void print_hello (string name ){
        cout<<"Hello "<<name<<endl;
   }
print_hello ("Tina")  //In main function
```

In the above example, function parameter is name. When you call print_hello() with an argument in the parentheses, the argument is assigned to the name parameter and the code in the function is executed. Inside the print() function call are some strings and the name variable. The output would be “Hello Tina”

**Question**: Write a program that takes 2 numbers as input from the user. Define a function to return sum of these 2 numbers.

```
#include <iostream>
using namespace std;
int addition(int number1,int number2)
{
  return number1+number2;
  //write your code here
}

int main()
{
  int num1,num2,a;
  cin>>num1>>num2;
  a = addition(num1, num2);
  cout<<a;
  return 0;
}
```

**The return statement**

A function is often used to return a value using a return statement. A return statement appears only inside the block where a function is defined. We need to assign the return type(the data type of data returned by function)

For example, a function to multiply two numbers with a return statement can be defined as

```
int multiplication(int x,int y ){
   int product = x * y;
    return product;
}
//Main function
int answer=multiplication(5,2);
cout<<answer<<endl;
```

Once the return statement executes, the program execution jumps immediately out of the block (just as the break statement makes the execution jump out of a while block). Any code after the return statement will not be executed. The program execution moves back to the line with the function call. The function call itself will evaluate to the function’s return value.

**Question**: Write a program that takes 2 numbers as input from the user. Define a function that returns smallest of these two numbers.

```
#include <iostream>
using namespace std;
int find_smallest(int num1,int num2)
{
  if(num1>num2){
    return num2;
  }else{
    return num1;
  }
```

```
  //write your code here
}
int main()
{
  int num1,num2;
  cin>>num1>>num2;
  cout<<find_smallest(num1,num2)<<endl;
  return 0;
}
```

**Variables and scope**

A variable that's inside the body of a function can't be used again when the function has finished execution. Because it exists only inside the function, this is called scope.

A local scope is created whenever a function is called. Any variables assigned in this function exist within the local scope. What makes variables in local scope special is that they are forgotten when the function returns and they will be re-created if the function is called again. The value of a local variable isn't remembered between function calls.

In the below function of finding the product of two numbers:-

```
int multiplication(int x,int y ){
int product = x * y;
return product;
}
//Main function
int answer=multiplication(5,2);
cout<<answer<<endl;
```

If we try to print the variable `product` outside the function definition, it will result in an error.

If a variable is defined outside the function, it has a different scope.

Variables that are assigned outside of functions exist in the global scope. There is only one global scope and it is created when your program begins. When your program terminates, the global scope is destroyed and all its variables are forgotten. Otherwise, the next time you run your program, the variables would remember their values from the last time you ran it.

A variable that exists in a local scope is called a local variable while a variable that exists in the global scope is called a global variable. A variable must be one or the other; it cannot be both local and global.

Local and global variables can have the same name but they are different variables because they are in different scopes.

For example,

```
int product = 100
int multiplication(int x,int y ){
int product = x * y;
return product;}
//Main function
int answer=multiplication(5,2);
cout<<answer<<endl;
```

The variable `product` that defined inside and outside function definition are not same.

Question: Define a global string variable **message** with value "This variable is in global scope". Define a function **print_message()** that displays "This variable is in local scope". Then in main function first print the variable **message,** then call the function **print_message().**

```
#include <iostream>
using namespace std;
string message = "This variable is in global scope";
```

```
void print_message() {
    string message = "This variable is in local scope";
    cout << message << endl;
}
int main() {
    cout << message << endl;
    print_message();
    return 0;
}
```

**Modules**

The C++ standard library provides numerous built-in functions that your program can call. For example, function strcat() to concatenate two strings, function memcpy() to copy one memory location to another location and many more functions.

#include<cstring> is used to include the required library.(Corresponding header file used in C is #include<string.h>)

For example:

```
strcat(str1,str2);
```

The above code will return concatenate(str1+str2) both strings and save in str1.

**Question**: Write a C++ program to take a name as input from user and say Hi . If the input is "Sam", then the output should be "Hi Sam".

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

int main()
{
  char name[10];
  char greet[4]="Hi ";
  cin>>name;
  strcat(greet,name);
  cout<<greet<<endl;

  return 0;
}
```

Write a program that takes an integer as input from the user. Define a function to display the square of that integer.

```
#include <iostream>
using namespace std;
void find_square(int x)
{
  cout<< x*x;
  //write your code here
}
int main()
{
  int num;
```

```
  cin>>num;
  find_square(num);
  return 0;
}
```

Write a program that takes two numbers as input from the user. Define a function that returns the greatest of the two given numbers. Then display that returned value.

```
#include <iostream>
using namespace std;
int find_largest(int num1, int num2)
{
  if(num1>num2){
    return num1;
  }else{
    return num2;
  }
  //write your code here
}
int main()
{
  int num1,num2;
  cin>>num1>>num2;
  cout<<find_largest(num1, num2);
  return 0;
}
```

Define a function to find square of a number. Define another function to find sum of two numbers. Use these functions to find the sum of squares of the two numbers entered by the user.

```
#include <iostream>
using namespace std;
int find_square(int num)
{
  return num*num;
  //write your code here
}
int find_sum(int num1, int num2)
{
  return num1+num2;
  //write your code here
}
int main()
{
  int num1,num2;
  cin>>num1>>num2;
  cout<<find_sum(find_square(num1), find_square(num2));
  return 0;
}
```

Write a program that takes two numbers as input from the user. Define a function that prints these values in descending order separated by a space.

```
#include <iostream>
using namespace std;
void arrange(int x,int y)
{
  if (x >= y) {
        cout <<x  << " " << y << endl;
    } else {
        cout << y << " " << x << endl;
  }
  //write your code here
}
int main()
{
  int num1,num2;
  cin>>num1>>num2;
  arrange(num1,num2);
  return 0;
}
```

Write a function to take the length of a side of a square as input from the user. Write functions to calculate and display area and perimeter of a square.
If input is 10, then output should look like:-
Area: 100
Perimeter: 40

```
#include <iostream>
using namespace std;
void area(int x)
{
  //write your code here
  cout<<"Area: "<<x*x<<endl;
}
void perimeter(int x)
{
  cout<<"Perimeter: "<<4*x;
  //write your code here
}
int main()
{
  int side;
  cin>>side;
  area(side);
  perimeter(side);
  return 0;
}
```

Write a function to check if the entered number is odd or even and print 'Odd' or 'Even' based on the input.

```
#include <iostream>
using namespace std;
string odd_or_even(int x)
{
  if(x%2==0){
    return "Even";
  }else{
    return "Odd";
  }
  //write your code here
}
int main()
{
  int num;
  cin>>num;
  cout<<odd_or_even(num);
  return 0;
}
```

Define a function to return factorial of a number entered by the user.

```
#include <iostream>
using namespace std;
int find_fact(int x)
{
  int fact=1;
  for(int i=1;i<=x;i++){
    fact*=i;
  }
  return fact;
  //write your code here
}
int main()
{
  int num;
  cin>>num;
  cout<<find_fact(num)<<endl;
  return 0;
}
```

Write a function to print the largest of three numbers entered by the user.

```
#include <iostream>
using namespace std;
void largest(int x,int y,int z)
{
  if(x>y && x>z){
    cout<<x;
```

```
  }else if(y>x && y>z){
    cout<<y;
  }else{
    cout<<z;
  }
  //write your code here
}
int main()
{
  int num1,num2,num3;
  cin>>num1>>num2>>num3;
  largest(num1,num2,num3);
  return 0;
}
```

You are given an array {4, 5, 6, 3, 7, 9}. Define a function that displays the sum of this array.

```
#include <iostream>
using namespace std;
void find_sum(int *arr,int n)
{
  int sum=0;
  for(int i=0;i<n;i++){
    sum+=arr[i];
  }
  cout<<sum;
  //write your code here
}
int main()
{
  int num[]={4, 5, 6, 3, 7, 9};
  int n = 6;
  find_sum(num,n);
  return 0;
}
```

You are given 3 numbers as input from the user. You have to display whether the first input lies in the range of second input and third input. For eg- If you are given the inputs 2, 0, 10. Then you have to check whether 2 lies in the range 0-10(Both inclusive).
You have to output "The number is in the given range." or "The number is outside the given range." based on the inputs.

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

void check_number(int n, int l, int u) {
    if (n >= l && n <= u) {
        cout << "The number is in the given range." << endl;
    } else {
        cout << "The number is outside the given range." << endl;
    }
```

```
}

int main() {
    int lower_limit, upper_limit, number;
    cin >> number >> lower_limit >> upper_limit;
    check_number(number, lower_limit, upper_limit);
    return 0;
}
```

Write a program that takes a number n as input. Define a function that returns the following value:-
f(n) = f(n-1) + 100, where f(0)=0

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

int calculate(int n) {
    if (n == 0) {
        return 0;
    } else {
        return calculate(n - 1) + 100;
    }
}

int main() {
    int num;
    cin >> num;
    cout<<calculate(num)<<endl;
    return 0;
}
```

You are given a sorted array arr[] = {3, 4, 6, 7, 8, 9, 11, 15, 18} and a number entered by the user. Using binary search, you have to find if the number is present in the array
The function should return the index position of the number if found, otherwise should print "Not found".

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

void binary_search(int *arr, int element) {
    int left = 0;
    int right = 8;
    int mid;

    while (left <= right) {
        mid = left + (right - left) / 2;

        if (arr[mid] == element) {
            cout <<mid<< endl;
            return;
```

```
        } else if (arr[mid] < element) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    cout << "Not found" << endl;
}

int main() {
    int arr[9] = {3, 4, 6, 7, 8, 9, 11, 15, 18};
    int num;
    cin >> num;
    binary_search(arr, num);
    return 0;
}
```

**Instruction**

Object oriented programming

In a procedural programming language like C, the entire code is written into one long procedure even though it might contain functions and subroutines. It is not manageable as both data and logic get mixed together. But when we talk about object-oriented programming, the program is split into self-contained objects or several mini-programs.
When working on complex programs in particular, objects are a way of organizing code in a program and breaking things down to make it easier to think about complex ideas. Object-oriented programming makes your code more readable, which in turn makes it more maintainable.
C++ is an object-oriented programming language. One of the most important concepts in object-oriented programming is the distinction between classes and objects, which are defined as follows:
Class --> A blueprint created by a programmer for an object. This defines a set of properties that will characterize any object. We can think of class as a way to classify objects into groups.
Object --> An instance of a class. This is the realized version of the class, where the class is manifested in the program.
For example: let's think of humans as a class which posses a set of properties like name age etc. then we all are different instances that are objects of the class human with different properties like name and age.
Question: _____ represents an entity in the real world with its identity and behaviour.

A A method

B An object

C A class

D An operator

**Methods and Attributes**

```
#include <iostream>
using namespace std;
class Lion{
public:
    void walk(){
        print("Lion is walking");
}
    void roar(){
        print("Lion is roaring");
}
int main()
{
Lion jake;
}
```

In this example, we initialized the object jake as an instance of the class Lion. We have defined access as public so it can be accessed from outside the class.
Let's use the two methods with the Lion object jake:

```
Lion jake;
jake.walk();
jake.roar();
```

And the output would be like,

```
Lion is walking
Lion is roaring
```

The class Lion using the two methods, walk() and roar(). We called these using the dot operator (.) which is used to reference an attribute of the object. In this case, the attribute is a method and it's called with parentheses.

**Question**: Define a class Computer with two methods system_in() and system_out(). The method system_in() should ask for the user's first name as input and system_out() should display the string "Thank you _____" (User's name in place of blank)
Create an instance of the class and call the methods.

```
#include <iostream>
using namespace std;
class Computer
{
  public:
  string name;
  void system_in(){
    cin>>name;
  }
  void system_out(){
    cout<<"Thank you "<<name;
  }
  //write your code here
};
int main()
{
  Computer sys1;
  sys1.system_in();
  sys1.system_out();
```

```
  return 0;
}
```

**Constructor**

The constructor is a member function used to initialize data. It is run as soon as an object of a class is created. It is defined using the same name as of class.

**There is no return type of constructor not even void.**

For example:

```
class Lion{
      Lion(){
              cout<<"This is the constructor method."<<endl;
          }
 };
```

If you added the above constructor to the Lion class in the previous example, the program would execute the print statement inside the constructor method, whenever you create a new object.

Let's create a constructor that passes parameters to assign variables to object.

```
#include <iostream>
using namespace std;
class Student{

public:
string first_name,last_name;
int roll_no;
//constructor
 Student(string first, string last, int roll){
 first_name = first;
 last_name = last;
roll_no = roll;
}

void exam(){
 cout<<"Exam registration number is:
"<<"2018"<<first_name<<last_name<<endl;
}

};
int main()
{
Student student1("Sophia", "John", 5);
cout<<student1.first_name<<endl;
 student1.exam();
 return 0;
}
```

The output would be,

```
Sophia
Exam registration number is: 2018SophiaJohn
```

Question: Create a class **Circle** with a data member **radius**.
Define a method to find the area of the circle.

Define a method to find the circumference of the circle.
Assume radius= 10 units
The output should look like:-
Area: 314
Circumference: 62.8

```
#include <iostream>
using namespace std;
class Circle
{
private:
    double r;
public:
    Circle(double radius) : r(radius) {}
    void area()
     {
        cout << "Area: " << 3.14 * r * r << endl;
    }

    void circumference()
     {
        cout << "Circumference: " << 2 * 3.14 * r << endl;
    }
};
int main()
{
  Circle circle1(10.0);
  circle1.area();
  circle1.circumference();
  return 0;
}
```

**Inheritance**

If a class is a part of another class, then it's a child of that class, and the other class is its parent. Classes can be both children of and parents to other classes.

```
class subclass_name : access_mode base_class_name
{
  //body of subclass
};
```

Inheritance is when a class uses code constructed within another class. If we think of inheritance in terms of biology, we can think of a child inheriting certain traits from their parent. That is, a child can inherit a parent's height or blood group. Children also may share the same last name with their parents.

Classes called child classes or subclasses inherit methods and variables from parent classes or base classes. We can think of a parent class called Parent that has class variables for last_name, height and blood group that the child class Child will inherit from the Parent. Because the Child subclass is inheriting from the Parent base class, the Child class can reuse the code of Parent allowing the programmer to use fewer lines of code and decrease redundancy.

For example:

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

class Fruits {
  public:
    Fruits()
    {
      cout << "This is a Fruit" << endl;
    }
};

class Apple: public Fruits{

};


int main()
{
    // creating object of sub class will invoke the constructor of base
classes
    Apple obj;
    return 0;
}
Output:
This is a Fruit
```

The private members in the base class cannot be directly accessed in the derived class while protected members can be directly accessed. For example, Classes B, C and D all contain the variables x, y and z in below example. It is just question of access.

```
class A
{
public:
    int a;
protected:
    int b;
private:
    int c;
};

class B : public A
{
    // a is public
    // b is protected
    // c is not accessible from B
};

class C : protected A
{
    // a is protected
    // b is protected
    // c is not accessible from C
};
```

```
class D : private A    // 'private' is default for classes
{
    // a is private
    // b is private
    // c is not accessible from D
};
```

**Types of inheritance in C++**

A derived class with only one base class is called **single inheritance.**

A derived class with one base class and that base class is a derived class of another is called **multilevel inheritance.**

A derived class with multiple base class is called **multiple inheritance.**

Multiple derived classes with same base class is called **hierarchical inheritance.**

**Question**:

Create a class **Person** with **name** as its data member. It should have a parameterized constructor that assigns a value to name.

Create another class **Student** which is a child class of Person. It should have **roll_no** and **gender** as its data members. Define a parameterized constructor that assigns values to name, roll_no and gender. Define a method **display()** to print the name, roll_no and gender respectively in separate lines.

In main function, create an instance of Student and assign the values (Sam,15,Male) to it using the constructor. Then display the details of the object by calling the display() method.

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

class Person {
protected:
    string name;

public:
    Person(string n) : name(n) {}
};

class Student : public Person {
private:
    int roll_no;
    string gender;

public:
    Student(string n, int roll, string gen) : Person(n), roll_no(roll),
gender(gen) {}

    void display() {
        cout << name << endl;
        cout << roll_no << endl;
        cout << gender << endl;
    }
};

int main() {
    Student student1("Sam", 15, "Male");
    student1.display();
```

```
    return 0;
}
```

## Defining classes and objects

A class is the blueprint from which specific objects are created.
A class is defined in C++ using keyword **class** followed by the name of class. The body of class is defined inside the curly brackets and terminated by a semicolon at the end.

```
class class_name
{
   //Access specifiers
   //Data Members
   //Member functions
};
```

### Accessing Data Members

Accessing a data member depends solely on the access control of that data member.
This access control is given by Access modifiers in C++. There are three access modifiers : public, private and protected.
Public - All the class members declared under public will be available to everyone. The data members and member functions declared public can be accessed by other classes too.
Protected - Protected access modifier is similar to that of private access modifiers, the difference is that the class member declared as Protected are inaccessible outside the class but they can be accessed by any subclass(derived class) of that class.
Private - The class members declared as private can be accessed only by the functions inside the class. They are not allowed to be accessed directly by any object or function outside the class
ClassName ObjectName;
Now creating a class is incomplete without some functionality. So functionalities can be defined by setting various attributes which acts as a container for data and functions related to those attributes.

```
#include <iostream>
using namespace std;
class example
{
   public:
   void method1()
   {
     // Print something
   }
};

int main() {
  example object1;
  object.method1();
   return 0;
}
```

Here a class named example is defined and a method is defined in class called method1. object1 is an object of class and method1 is called using object of the class.
Question: A class definition is given below, fill in the blanks with appropriate keywords

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

```
{
   public:
   void printname()
   {
     cout <<"Inside the Class"<<endl;
   }
};

int main() {

   _____ obj1;    // Blank 1
   _____.printname();  //Blank 2
   return 0;
}
```

A Blank 1: lpu
Blank 2: lpu

B Blank 1: int
Blank 2: obj1

C Blank 1: lpu
Blank 2: obj1

D Blank 1: string
Blank 2: lpu

---

You are given a Temperature class with celsius and farenh as data members. Define two member functions :
1. celsiusToFahren - It will take a temperature in celsius and will print its value in fahrenheit.
2. fahrenToCelsius - It will take a temperature in fahrenheit and will print its value in celsius.
The values should be printed correct to one decimal place.

```
#include <iostream>
#include<iomanip>
using namespace std;
class Temperature
{
  public:
  float celsius,farenh;

  void celsiusToFahren(float celsius1)
  {
   farenh = (celsius1 * 9/5) + 32;
    cout<<fixed<<setprecision(1)<<farenh<<endl;
    //write your code here

  }
  void fahrenToCelsius(float farenh1)
```

```
  {
   celsius = (farenh1 - 32) * 5/9;
    cout<<fixed<<setprecision(1)<<celsius<<endl;
    //write your code here

  }
};
int main()
{
        Temperature temp;
        temp.celsiusToFahren(10);
        temp.fahrenToCelsius(100);
    return 0;
}
```

Create a class Student with data members name, roll_no and marks. Define the following methods :-
1. A parameterized constructor - It should take a string and an integer as arguments and assign them to name and roll_no.
2. setMarks() - It should assign marks to the student.
3. display() - It displays the information about the student in the format given in the expected output.

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

class Student
{
  public:
  string name;
  int roll_no,marks;
      Student(string name1, int roll_no1) : name(name1), roll_no(roll_no1) {}
  void display_info()
  {
    cout<<"Name: "<<name<<endl;
    cout<<"Roll no: "<<roll_no<<endl;
    cout<<"Marks: "<<marks<<endl;
  }
  void setMarks(int marks1)
  {
    marks = marks1;
    //write your code here
  }
};
int main()
{
  Student student1("Emma", 18);
  student1.setMarks(68);
  student1.display_info();
  return 0;
```

```
}
```

You are given a class Date that has data members day, month and year. It has a method print_day() to display the date.
You have to write its parameterised constructor that takes 3 arguments and assign these to its data members.

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

class Date
{
public:
    string date, month, year;

    Date(string d, string m, string y) : date(d), month(m), year(y) {}

    void print_day()
     {
        cout << date << " " << month << " " << year << endl;
    }
};

int main()
{
    Date name("08", "April", "2000");
    name.print_day();
    return 0;
}
```

Create a class Fruits with data members name, price and quantity. Define the following methods :
-
1. A parameterized constructor that takes 2 arguments and assigns these values to name and price.
2. setQuan() - It assigns quantity to the object.
3. total() - It calculates the total price and then displays it.

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

class Fruits
{
public:
    string name;
    int price;
    int quantity;

    Fruits(string n, int p) : name(n), price(p) {}

    void setQuan(int q)
     {
```

```
        quantity = q;
    }

    void total()
     {
        int totalPrice = price * quantity;
        cout << totalPrice << endl;
    }
};

int main()
{
    string n;
    int p, q;
    cin >> n >> p >> q;

    Fruits fruit(n, p);
    fruit.setQuan(q);
    fruit.total();

    return 0;
}
```

Create a class Animals. It has method eating() that prints 'Eating....' .
There are two subclasses Herbivores and Carnivores.
Define method for eating in each subclass that first call the eat method of the base class and then print 'Grass' in case of Herbivores and 'Meat' in case of Carnivores.

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

class Animals
{
public:
    void eat()
     {
        cout<<"Eating...."<<endl;
    }
};

class Herbivores: public Animals
{
public:
    void herb_eat()
     {
        eat();
        cout<<"Grass"<<endl;
    }
};

class Carnivores: public Animals
```

```
{
public:
    void carn_eat()
     {
        eat();
        cout<<"Meat"<<endl;
    }
};

int main()
{
    Herbivores cow;
    cow.herb_eat();

    Carnivores lion;
    lion.carn_eat();

    return 0;
}
```

Create a class Rectangle which has data members length and width. You have to define a constructor which takes 2 arguments and assign these to its data members. Also define a function area() which prints the area of the rectangle.

```
#include <iostream>
using namespace std;
class Rectangle
{
public:
    int length, width;
    Rectangle(int l, int w) : length(l), width(w) {}
    void findArea()
     {
        int area = length * width;
        cout <<area << endl;
    }
};

int main()
{
    int l, w;
    cin >> l >> w;
    Rectangle rect1(l, w);
    rect1.findArea();
    return 0;
}
```

Create a class Shape and its subclass Square. Square has a data member length. You have to define a constructor of Square class which takes length as an argument and assigns it to length. You also have to define the area functions which can find the area of the shape. The area function

in the Shape class returns the value 0. The area function in the Square class returns the area of the square.

```
#include <iostream>
using namespace std;
class Shape
{
public:
    virtual int area()
     {
        return 0;
    }
};
class Square : public Shape
{
public:
    int length;
    Square(int l) : length(l) {}
    int area() override
     {
        return length * length;
    }
};
int main()
{
    int n;
    cin >> n;
    Square sq(n);
    cout << sq.area() << endl;
    return 0;
}
```

Define a class Person and its two child classes: Male and Female. All classes have a method getGender() which can return "Male" for Male class, "Female" for Female class and "Unknown" for Person.

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

class Person
{
public:
    virtual string getGender()
     {
        return "Unknown";
    }
};

class Male : public Person
{
public:
```

```
    string getGender() override
     {
        return "Male";
    }
};

class Female : public Person
{
public:
    string getGender() override
     {
        return "Female";
    }
};

int main()
{
    Male male_obj;
    Female female_obj;

    cout << male_obj.getGender() << endl;
    cout << female_obj.getGender() << endl;

    return 0;
}
```

You are given a class Poem. It has a string array as its data member. It has a display function that displays the elements in the array in separate lines. You have to define its constructor that assigns each line of the poem below to a different element of the string array.
When I behold a forest spread
With silken trees upon thy head;
And when I see that other dress
Of flowers set in comeliness;

```
#include <iostream>
using namespace std;
class Poem
{
public:
    string poem_arr[4];
    Poem()
    {
        poem_arr[0] = "When I behold a forest spread";
        poem_arr[1] = "With silken trees upon thy head;";
        poem_arr[2] = "And when I see that other dress";
        poem_arr[3] = "Of flowers set in comeliness;";
    }
    void getPoem()
     {
        for (int i = 0; i < 4; i++)
        {
```

```
            cout << poem_arr[i] << endl;
        }
    }
};
int main()
{
    Poem obj;
    obj.getPoem();
    return 0;
}
```