Important Short Questions

Q2. Give short answers of the following questions.

Q1. Define function.

Ans: A function is a group of statements that together perform a task.

Q2. Write down the purposes of functions in C++

Ans: Every C++ program has at least one function which is main (). Additional functions can be defined in the program. Each function performs a specific task. Functions are one of the main building blocks of any programming language. Functions are used to provide modularity to a program. Creating an application using function makes it easier to understand edit, check errors etc. Functions make the work easier by writing the code once and executing it again and again as many times as required.

Q3. Name the types of functions.

Ans: C++ functions are categorized into the following two types

(i) Library or built-in functions

(ii) User defined functions

Q4. Define built-in functions.

Ans: Built-in functions are the pre-defined functions in C++

Q5. Write down the uses of built-in functions,

Ans: Programmers can use these function by invoking calling them directly. They do not need to write code for them.

Q6. Describe the use of built-in functions in various fields.

Ans: C++ provides a series of library or pre-defined functions for various commonly used operations of algebra, geometry, trigonometry, finance, graphics, clipboard etc.

Q7. Which header file must be included to use built-in functions?

Ans: Header file <cmath> must be included in the program to use built-in functions.

Q8. Give five examples of built-in functions

Ans: Examples of commonly used built-in functions are

(i) Abs( )
(ii) Div( )
(iii) Getchar( )
(iv) Log( )
(v) Pow( )

Q9. What is meant by user-defined function?

Ans: C++ allows programmers to define their own functions. The functions defined by the programmers according to their needs are called user-defined functions.

Q10. State the syntax to define user-defined function.

Ans: The general syntax for defining a user-defined function is

Return type function-name (parameters)

{

//function-body

}

Q11. Write down any two advantages of using functions

Ans:

1. Using functions a program can be defined in logical blocks it will make the code clear and easy to understand
2. Use of function avoids typing same pieced of code multiple times. User can call a function to execute same line of code multiple times without re-writing it.

Q12. Name the parts of function signature.

Ans: the signature of a function comprises of the parts given below

1. Name of the function
2. Arguments
3. Return type of function

Q13. How many components of a function are there? Write their names.

Ans: There are three components

1. Function prototype/declaration
2. Function definition
3. Function call

Q14. What is meant by function prototype/declaration?

Ans: function prototype/declaration is done to tell the compiler about the existence of the function. Function's return type its name and parameter list is mentioned. A function prototype/declaration is used before the main ( ) function. It ends with a semicolon (;)

Q15. State the use of function prototype.

Ans: function prototype is used in C++ program only when the function is defined after the definition of main ( ) function. If the function definition lies before the main ( ) function then there is no need for the function prototype.

Q16. What is meant by function definition?

Ans: A function definition specifies what a function does.

Q17. How many parts of function definition are there?

Ans: There are two parts

(1) Header

(2) Function body enclosed in { }

Q18. Define function header.

Ans: Function header is similar to function prototype with the only difference that it has the variables name for the parameters and no semicolon (:)

Q19. What is meant by function call?

Ans: A function call is a statement in the calling function (e.g main ( ) function) to execute the code of the function.

Q20. What is meant by scope of variable?

Ans: by the scope of a variable we mean that in which parts of the same program the variable can be accessed

Q21. How many scopes of variables are there?

Ans: there are three scopes of variables

(a) Local scope
(b) Global scope
(c) Static scope

Q22. What is meant by local/automatic variables?

Ans: Variables defined within a block of a function are called local variables.

Q23. What is meant by global variable?

Ans: Variables defined at the top of a program before the main ( ) function are called global variables.

Q24. How do we declare global variables?

Ans: In order to declare global variables we simply have to declare the variable outside any function or block that means directly in the body of the program.

Q25. Define Static variables.

Ans: Static variables are those variables which are preceded by the keyword static wile declaring them.

Q26. What is meant by initialization of static variables?

Ans: static variables are initialized once in the program and remains in memory until the end of the program. These variables have the capability to preserve information about the last value of function returned. Static variables are local in scope to their module or function which they are defined.

Q27. What is meant by parameters?

Ans: In function prototype and function call the variables and values (written in the parenthesis) are called parameters.

Q28. How many types of parameters are there? Name them.

Ans: There are two types of parameters:

1) Formal parameters
2) Actual parameters

Q29. What are formal parameters?

Ans: Formal parameters are those parameters which appear in function declaration/header and also in function prototype.

Q30. Define actual parameters.

Ans: Actual parameters are those parameters which appear in function cells.

Q31. Name the types of functions based on their scope.

Ans: Based on the scope of the function. Functions are categorized into two types.

a) Local functions
b) Global functions

Q32. What are local functions?

Ans: Those functions which are defined inside the body of another function are called local functions.

Q33. Where do we use built-in function in program?

Ans: Normally we use built-in function inside the body of main ( ) function to perform our activities such declarations are termed as local.

Q34. Define global function.

Ans: A function declared outside any function is called global function.

Q35. How to access global function?

Ans: A global function can be accessed from any part of the program. Normally, user-defined functions are considered as global function because usually they are defined before the main ( ) function and are thus accessible to every part of the program.

Q36. Why do we need inline function?

Ans: Using function calls in program a lot of CPU time is wastes in passing control from the calling program to the called function and returning back to the calling program. This limitation can be overcome by the use of inline function.

Q37. How does inline function work?

Ans: In inline function the function return type is preceded by the inline keyword which requests the compiler to treat the function as an inline and do not jump again and again to the calling function and back to it. In the case of inline function when the compiler compiles the code, all inline functions are expanded in-place. That is the function call is replaced with a copy of the contents of the function itself, which removes the function call overhead.

Q38. Write down the disadvantage of the inline function

Ans: The disadvantage of the inline function is that it can make the complied code quite larger especially if the inline function is long and/or there are many calls to the inline function.

Q39. Describe the format for the declaration of inline function.

Ans: The format for the declaration of inline function is given in the following segment

```
//inline function declaration
Inline type name (arguments....)
{
Statements;
}
```

Q40. Why do we need arguments in programming?

Ans: When we want to execute functions we need to pass arguments to them. The results produced within the function body is then returned back to the calling program.

Q41. How many methods of passing arguments to functions are there?

Ans: The following are the most commonly used methods of passing arguments to functions

1) Passing arguments by constants
2) Passing arguments by values
3) Passing arguments by reference

Q42. What is meant by passing arguments by constants?

Ans: While calling a function arguments are passed to the calling function. In passing arguments by constants the actual constant values are passed instead of passing the variables holding these constants.

Q43. What is meant by passing arguments by values?

Ans: By default, arguments in C++ are passed by value. When arguments are passed by value, a copy of the argument is passed to the function.

Q44. What is meant by passing arguments by reference?

Ans: In pass by reference, the reference to the function parameters is passed as arguments rather than variables. Using pass by reference, the value of arguments can be changes within the function. In passing arguments by reference, the original variables should precede by the ampersand sign (&).

Q45. How values are returned by function?

Ans: Sometimes we need a function to return multiple values. However, functions can only have one return value. One way to return multiple values is the use of the method of passing arguments by reference.

Q46. Write down the syntax of default arguments.

Ans: The syntax of default arguments is given below

Type function_name(parameter1=value,....);

Q47. What is the use of return statement?

Ans: If the return type of a function is any valid data type such as int, long, float, double, long double or char then return statement should be used in the function body to return the result to the calling program (main ( ) function).

Q48. Write down the Syntax of return statement.

Ans: The general syntax of return statement is given here under.

Return (expression of variable holding results or constant) :

Q49. What is meant by function overloading?

Ans: Function overloading is a feature of C++ that allows to create multiple functions with the same name so long as they have different number of types of parameters.

Q50. Write down any two advantages of function overloading.

Ans: a) using function overloading, we can declare multiple functions with the same name that have slightly different purposes

b) Function overloading can significantly lower complexity of programs

Q51. Write down the features of function overloading.

Ans: The features of function overloading are listed here under

a) Number of arguments
b) Data types of arguments
c) Return type

Q52. What is meant by return type of a function?

Ans: The return type of a function is not considered when functions are overloaded. It means that if two or more functions with the same name have same function signatures but different return types then they are not considered as overloaded and the compiler will generate error message.

Q53. Write a C++ program to add two integers. Make a function add( ) to add integers and display sum in main ( ) function.

Ans:

```
#include <iostream>
Using namespace std;
//Function prototype (declaration)
Int add(int , int);
Int main ( )
{
Int num1, num2, sum;
Cout < < "Enters two numbers to add:" ;
Cin > > num1 > > num2;
//Function call
Sum=add (num1, num2);
Cout < < "Sum=" < < sum;
Return 0;
}
//Function definition
Int add(int a, int b)
{
Int add;
Add=a+b;
//Return statement
```

```
Return add:
}
```

Q54. Write a C++ program to check prime number by making user-defined function. In the program, no arguments will pass and will not return value.

Ans:

```
#include <iostream>
Using namespace std;
Void prime( );
Int main( )
{
//No argument is passed to prime( )
Prime( ) ;
Return 0;
}
//Return type of function is void because value is not returned
Void prime( )
{
Int num, I, flag=0;
Cout < < "Enter a positive integer enter to check:";
Cin > > num;
For (i=2; i<=num/2; ++1)
{
```

```
If(num % I ==0)
{
Flag=1:
Break;
}
}
If (flag==1)
{
Cout < < num < <"is not a prime number";
}
Else
{
Cout < < num < < "is a prime number":
}
}
```

Q55. Write a C++ program to check prime number by making a user-defined function. In the program, no arguments will be passed but will return a value.

Ans:

```
#include <iostream>
Using namespace std;
Int prime( ) :
Int main ( )
```

```
{
Int num, i, flag=0;
//No argument is passed to prime ( )
For (i=2; i<=num/2; ++i)
{
If (num%i==0)
{
Flag=1;
Break;
}
}
If (flag==1)
{
Cout <<num<<"is not a prime number";
}
Else
Cout <<num<<"is a prime number";
}
Return 0;
}
//Return type of function is int
Int prime ( )
{
```

```
Int n;
Printf("Enter a positive integer to check: ");
Cin >> n;
Return n ;
}
```

Q56. Write a C++ program to check prime number by making user-defined function. In the program, arguments will pass but will return no value.

Ans:

```
#include <iostream>
Using namespace std;
Void prime (int n);
Int main ( )
{
Int num;
Cout < < "Enter a positive integer to check:";
Cin >> num;
//Argument num is passed to function prime( )
Prime(num);
Return 0;
}
//There is no return value to calling function. Hence, return type of function is
void
```

```
Void prime(int n)
{
Int i, flag=0:
For (i=2, <=n/2: ++i)
{
If (n%i==0)
{
Flag=1:
Break;
}
}
If (flag==1)
{
Cout < < n < <" is not a prime number";
}
Else {
Cout < < n < <"is a prime number";
}
}
```

Q57. Write a C++ program to check prime number by making user-defined function. In the program, arguments will passed will return a value.

Ans:

```
#include <iostream>
Using namespace std;
Int prime(int n);
Int main ( )
{
Int num, flag=0;
Cout < < "Enter positive integer to check:";
Cin > > num;
//Argument num is passed to check ( ) function
Flag=prime(num);
If (flag==1)
Cout < < num < < "is not a prime number";
Else
Cout < < num < < "is a prime number";
Return 0;
}
//This function returns integer value//
Int prime(int n)
{
Int i;
For (i=2 i<=n/2; ++i)
{
If(n%i==0)
```

```
Return 1;
}
Return 0;
}
```

Q59. Write a C++ program to compute absolute value for both integer and float

Ans:

```
#include <iostream>
Using namespace std;
Int absolute(int);
Float absolute(float);
Int main( )
{
Int a=5;
Float b=5.5;
Cout < < "Absolute value of " < < a < < "=" < < absolute (a) < < end1;
Cout < < "Absolute value of " < < b < < "=" < < absolute (b);
Return 0;
}
Int absolute (int var)
{
If (var < 0)
Var= -var;
```

```
Return var:
}
Float absolute (float var)
If (var < 0.0)
Var=-var;
Return var:
}
```

Q60. Write a C++ program to demonstrate local variables.

Ans:

```
#include <iostream>
Using namespace std;
Void test ( );
Int main ( )
{
//local variable to main ( )
Int var=5;
Test ( )
//illegal varl not declared inside main ( )
Varl=9;
}
Void test ( )
{
```

```
//local variable to test ( )
Int varl;
Varl=6;
//illegal var not declared inside test ( )
Cout < < var;
}
```

Q61. Write a C++ program to demonstrate global variables.

Ans:

```
#include <iostream>
Using namespace std;
//Global variable declaration
Int c=12;
Void test ( ) :
Int main ( )
{
++c;
//Outputs 13
Cout < < c < < end1;
Test ( )
Return 0;
}
Void test ( )
```

```
{
++C:
//Outputs 14
Cout < < c;
}
```

Q62. Write a C++ program to explain the concept of statistical local variable.

Ans:

```
#include <iostream>
Using namespace std:
Void test ( )
{
//var is a static variable
Static int var=0:
++var:
Cout < < var < < end1:
}
Int main ( )
{
Test ( ):
Test ( ):
Return 0:
}
```

Q63. Write a C++ program to sort two numbers using call by reference. First, display smallest number

Ans Program

```
#include <iostream>

Using namespace std;

//Function prototype for call by reference

Void swap(float &x, float &y);

Int main ( )

{

Float a, b;

Cout<<"Enter 2 numbers:"<<end1;

Cin>>a>>b;

If (a>b)

Swap(a,b);

//Variable a contains value of smallest number

Cout<<"Sorted numbers:";

Cout<<a<<" "<<b<<end1;

Return 0;

}

//A function definition for call by reference

//The variables x and y will have their values changed

Void swap(float &x, float &y)

//Swaps x and y data of calling function
```

```
{
Float temp;
Temp=x;
X=y;
Y=temp;
}
```

Q64. Write a C++ program to calculate the area of a regular hexagon inscribed in a circle of radius input by the user

Ans. Program

```
#include <iostream>
Using namespace std;
//AreaTriangle function prototype
Float AreaTriangle (float base, float height);
Int main ( )
{
Float side, segmentHeight, hexagonArea;
Float cosTheta=0.866025;
Cout<<"Program to calculate the area of a hexagon"<<end1;
Cout<<"Enter side of hexagon:";
Cin>>side;
//Base of triangle is equal to side, height is side*cos(30)
segmentHeight=side*cosTheta;
```

```
//Function returns area of segment. 6 segments for total area.
hexagonArea=6.0*AreaTriangle(side,segmentHeight);
cout<<"Area of hexagon="<<hexagonArea<<end1;
return0;
}
//AreaTriangle function definition
Float AreaTriangle (float base, float height)
{
Float area;
Area=(base*height)/2.0;
Return area;
}
```

Q65. Write a C++ program to calculate factorial of a number with function call.

Ans. Program

```
#include <iostream>
Using namespace std;
//Function declaration (prototype)
Int Factorial (int M);
Int main ( )
{
Int number=0; result;
//User input must be an integer number between 1 and 10
```

```
While (number<1 | | number>10)
{
Cout<<"Integer number=":
Cin>>number;
}
//Function call and assignment of return value to result
Result=Factorial (number):
//Output result
Cout<<"Factorial=" <<result<<end1;
Return 0;
}
//Function definition (header and body)
//An integer, M, is passed from caller function
Int factorial (int M)
{
Int i, factorial=1:
//Calculate the factorial with a FOR loop
For (i=1; i<=M; ++i)
{
Factorial=factorial*I;
}
Return factorial://This value is returned to caller
}
```

Q66. Write a C++ program to calculate the factorial (n!) of an integer number (n) using a FOR loop to compute

N!=1*2*3...(n-2)*(n-1)*n

Ans. Program

```
#include <iostream>
Using namespace std;
Int main ( )
{
Int I, number=0, factorial=1;
//User input must be an integer number between 1 and 10
While (number<1 | | number>10)
{
Cout<<"Enter integer number (1-10)=";
Cin>>number;
}
//Calculate the factorial with a FOR loop
For (i=1; i<=number; ++i)
{
Factorial=factorial*I;
}
//Output result
Cout<<"Factorial=" <<factorial<<end1;
Return 0;
```

```
}
```

Q67. Write a C++ program to display greeting message by using function

Ans. Program

```
#include <iostream.h>
#include <conio.h>
Void greeting(void);
Void main ( )
{
Clrscr( ):
Char ch;
Cout<<"Press y for greeting message:";
Cin>>ch;
If (ch=='y' | | ch=='y')
{
Greeting( ):
}
Cout<<"\n\nPress any key to exit...!!\n";
Getch( ):
}
Void greeting(void)
{
Cout<<"\nAssalam-o-Alykum, Programmer!";
```

```
Cout<<"\nWelcome to C++ Functions":
}
```

Q68. Write a C++ program that asks the user to enter a number to find its cube using the function and then display the cube of that number on the output.

Ans. Program

```
#include <iostream.h>
#include <conio.h>
Float cube(float):
Void main ( )
{
Clrscr( ):
Float numpass, numget:
Cout<<"Enter a number to calculate its cube:\n":
Cin>>numpass:
Numget=cube(numpass):
Cout<<"\nThe cube of"<<numpass<<" is "<<numget:
Getch( ):
}
Float cube(float x)
{
Float temp:
Temp=x*x*x;
```

```
Return temp;
}
```

Q69. Write a C++ program to swap two values by call by reference method

Ans. Program

```
#include <iostream.h>
#include <conio.h>
Void swap (int &, int&);
Void main ( )
{
Clrscr( );
Int a, b;
Cout<<"Enter any two number:";
Cin>>a>>b;
Cout<<"\nThe original values are:\n";
Cout<<"a="<<a<<"\tb="<<b<<"\n";
Swap (a,b);
Cout<<"\nThe values after swap( ) are:\n";
Cout<<"a="<<a<<"\tb="<<b<<"\n";
Getch( );
}
Void swap(int &x, int &y)
{
```

```
Int temp;
Temp=x;
X=y;
Y=temp;
Cout<<"\nThe swapped values are:\n";
Cout<<"a="<<x<<"\tb="<<y<<"\n";
}
Void swap(int &x, int &y)
{
Int temp;
Temp=x;
X=y;
Y=temp;
Cout<<"\nThe swapped values are:\n";
Cout<<a"="<<x<<"\tb="<<y<<"\n";
}
```