SMVS
Page No.
Date : / /

1 Write down the benefits of Object Oriented Programming.

This are the main benefits of Object Oriented Programming.

1 Object Oriented Programming is faster and easier to execute.

2 OOP Provides a clear structure for the Program.

3 OOP makes the code ~~easir~~ easier to maintain, modify and debug.

4 OOP ~~make~~ take less code time and shorter development time.

5 OOP is allows to break the program into the bit-sized problem that can be solved easily.

6 OOP Systems can be easily upgraded from small to large system.

7 The principle of data hiding helps the programmer to build secure program.

8 By using inheritance, we can eliminate redundant code and use in onther class.

9 It is possible that multiple instances of object co-exist without any interference.

10 In OOP, we can add new data and function very easy.

11 Object oriented programming have specifiers like private, public etc.

12 Using abstraction, we can display only essential information and hiding the details.

13 Using Encapsulation, we can wrapping data and information under a single unit.

2 Explain call by value and call by reference concept with suitable example.

=> Call by Value:

In this method, values of actual parameters are copied into function's formal parameters.

While calling a function, we pass values of variable that is ~~call~~ known as call by value

Values of variable are passes by simple technique.

Example:

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

int swap (int &x, int &y)
{
  int temp;
  temp = x;
    x = y;
```

```
    Y = temp;
}
int main()
{
    int X = 20, Y = 30;
    swap(X, Y);

    cout << X << endl;
    cout << Y;

    return 0;
}
```

=> Call by reference:

In call by reference, while calling a function, instead of passing the values of variable than we pass address of variable to the function it is known as "call by reference."

In this method, the address of actual variables in the calling function.

Pointer variables are necessary to define to store the address values of variables.

Example :

```
#include <iostream>
using namespace std;
int main()
{  int a = 8;
   int &b = a;

   cout<<"The variable a:"<<a;
   cout<<"The variable r:"<<b;

   return 0;
}
```

SMVS
Page No.
Date : / /

3 Explain function with default argument passing with suitable example.

A default argument is a value provided in a function declaration that is automatically assigned by the compiler.

Function with default argument is reducing the size of program.

Default arguments improves consistency of program

-> Syntax:

return data type Function name (Default argument) ;

-> The default argument is an assigned value in the function declaration.

→ Example:

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

int sum(int x, int y)
{
    return (x+y);
}

int main()
{
    cout << sum(10, 15) << endl;

    return 0;
}
```

4 Write a structure of C++ program and explain in brief.

A structure is usually defined at the beginning of a program.

Structure of C++ Program

Header	#include <iostream>
Namespace	using namespace std;
main()	int main()
Body	cout << "a" << endl;
Return	return 0;

- Header Files : The first component is the header files in c++ program.

In c++ program iostream header files is use.

These is use standaard input, output and error facillitles are provided.

These standard stream process data as a stream of characters and data is read and displayed in a continuous flow.

This standard stream defined in <iostream>.

- Namespace : A namespace permits grouping of various entities like classes, objects, token etc.

The c++ std standards commitee has rearranged the entities of the standards library under a namespace called std.

It is defined -> Using namespace std;

- Main Function: The main() is a startup function that starts the execution of program.

- All the statements that need to be execution are written only main function.

- Body: In body function refers to the operations that are performed in the function.

Ex. cout << "a" << endl;
cin >> a;

-> cout: It is used to be displayed data on a screen.

-> cin: It is used to read input data on a screen.

• Return:

The last part of the C++ program is return statement.

If the return type is int then will be return 0 is return statement.

5 Explain the concept of class and objects.

=> Class:

Class in C++ is the building block that leads to Object-Oriented programming.

Class is a user-defined data type.

Class holds its own data members and member function.

When we created class, there is no memory allocated to the class.

-> Syntax :

```
class Classname
{
  Access specifier:
  Data members;
  Member function()
};
```

-> There is only one way to define a class that is using class keyword.

Class is a group of similar objects.

Class is ~~decla~~ declared only once.

Class is declared using class keyword.

=> Object:

Object is an instance of a class.

When defined object, than memory is allocated to the class.

Object is created many time as per requirement of user.

When Object is created, than memory is allocate for it.

-> Syntax:

```
classname objectname;
```

=> Example:

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

  class student
{
  Public :
```

```
    string name;

    void display()
    {
    cout<<"Khushi"<<endl;
    }
};

int main()
{ student ob1;

    ob1.student;
    ob1.display();

    return 0;

}
```

6 Explain constructor in derive class with suitable example.

Constructor is a special method that is automatically called when object is created.

Constructor has same name of class.

Constructor is called atomaticaly by a complier.

Constructor does not return any type of value.

-> There are Three types of Constructor.
1) Default constructor
2) Parametrizad constructor
3) Copy constructor.

1 Default Constructor : Default Constructor is also know as a no argument constructor.

Default constructor is used for function is called by default argument.

2 Parametrizad Constructor:

The constructor which has parameter or argument it is called parametrizad constructor.

Sometime it is required different data element of object which has different argument.

3 Copy constructor:

In copy constructor, it use to declare one object to using anthor object.

Copy Constructor is a type of Parametrizad constructor which accept our argument to its own class in parameter.

-> Example:

```
class demo
{
  int x, y;
  Public:

    demo(int m, int n) (Parametrizad Constructor)
    {
        x = m;
        y = n;
    }
    void print()
    {
      cout<<"x"<<x;
      cout<<"y"<<y;
    }
    demo(demo &a) (Copy constructor)
    {
        x = a.x;
        y = a.y;
    }
};

int main()

{ demo a(1,2)
```

```
a.demo();
demo b
  b = a;
b.print();

  return 0;
}
```

(Default Constructor)

18 Explain Destructor with an example.

object that are created by constructor.

Destructor cannot to ~~be~~ declared static or const.

Destructor does not have any arguments.

A Destructor should be declared in the k Public section of class.

Destructor has no return type not even void.

-> Syntax :

~ constuctor name ()

-> Destructor name is name as a constuctor name.

-> Example :

```
Class demo
{
  int x;
  Public:

    demo ()
    { x = 0;
    }

    void print ()
   {
    cout << " x " << x;
   }
```

```
~demo()
{
    cout << "Destructor" << endl;
        called
    }
};

int main()
{
    demo a;

    a.print();

    return 0;
}
```

15 Write program in c++ to use multiple constructor.

```
class demo
{
 int x, y;
 Public:
   demo(int m, int n)
   {
     x = m;
     y = n;
   }
   void print()
   {
    cout<<"x"<<x;
    cout<<"y"<<x;
   }
   demo(demo &a)
   {
      x = a.x;
      y = a.y;
   }

};
```

```
int main ()
{
   demo a (1, 2)
    a.demo ();
   demo b
      b = a;
   b. print ()

   return 0;

}
```

9 Explain Type Conversion in detail.

Type conversion is converted into anthor data type by a complier.

Type conversion is no needed casting operator.

Type conversion is widening conversion.

Type conversion data type is no smaller than source data type.

Type conversion is less use in coding and competitive program.

Type conversion is less effient and less reliable.

There are two type of Type Conversion.
1) Implicit
2) Explicit.

1 Implicit Type Conversion:

Implicit conversion is also known as 'Automatic Type Conversion'.

This conversion is done by the compiler on its own.

This conversion is generally use at take place when in an expression more than one data type is present.

2 Explicit Type Conversion:

This conversion is user defined conversion.

Explicit type conversion is done by the user ⇔ by using operator.

10 Explain user defined datatypes of C++.

The data types that are defined by the user that is called user defined datatypes.

User defined datatypes is easier to build and easier to verify.

There are many type of user defined datatype.

1 Class: Class is also user defined datatype.

Class is holds its own data member and member function in program.

-> Syntax:

```
class Classname
{
  Access Specifier:
  Data member;
  Member function()
};
```

2 Structure: Structure is also user defined datatype in program.

A structure creates a data type that can be used to group items of program.

-> Syntax:

```
struct address
{
```

```
char name[50];
char street[50];
char state[20];
int pin;
};
```

3 Union: Union is also user defined datatype in program.

In union, all members share the same memory location.

4 Enumeration: Enumeration is a also user defined datatype.

Enumeration is mainly used to assign names to integral constant

11 Explain Function Prototyping with example.

Prototype is a declaration in c++ of a function, its name, parameters and return type before its actual declaration

Prototype is tells the return type of the data that the function return.

Prototype is tells the number of arguments passed to the function.

Prototypes tells the data type of each of the passed arguments.

Prototypes also tell the order in which the arguments are passed in the function.

-> Example:

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

 int add (int a, int b)
```

```
{
    return (a+b);
}

int main()
{
    int sum;

    sum = add(100, 10);

    return 0;

}
```

-> In this example function prototype is int add (int, int);

This provides the compiler with information about the function name and its parameters.

SMVS
Page No.
Date : / /

12 Describe Friend function with example.

Friend function can be access in class to use of private and protected members.

Friend function is usefull to allow a prticular class to access private members of other class.

Friend function is not mutual for both the class.

Friend function is not inherited

Friend function are declared in one or more class in program.

Friend function is no part of class in program.

-> Syntax:

Return type Function name (class name object)

-> example:

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

class M2;
class M1
{
  int X;
  Public:
    void get value ()
    {
      cout << "Enter value" << X;
      cin >> X;
    }

    Friend void multiply (M1, M2);
};

class M2
{
  int Y;
  Public:

    Void getvalue()
```

```
{cout << "Enter value" << y;
 cin >> y;
}

friend void multiply (M1, M2);

};

Void multiply (M1 a, M2 b)
{
  int mul;
  mul = a.x * b.y;
  cout << "Multiplication " << mul << endl;
}

int main()
{
  M1 a;
  a.getvalue();
  M2 b;
  b.getvalue();

multiply(a, b);

return 0;

}
```

13 Explain Function Overloading in detail.

Function Overloading is a feature of Object oriented Programming.

In Function overloading, we can use two or more same name function with different parameters.

In Function overloading, function name should be same and argument should be different.

Function overloading is that it improve code reudabillity and allows code reusability.

The use of function overloading is ~~sam~~ save memory space in program.

Same function perform ~~diffe~~rent operation with different argument.

-> Example:

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

void print(int i)
{
  cout << " Int i is " << i << endl;
}

void print(double f)
{
  cout << " Float f is " << f << endl;
}

int main()
{
  print(10);
  print(10.10);

  return 0;
}
```

#

16 Explain the usage of inline function with a suitable example.

Function defined inside the class body in a single line that is called inline function.

c++ provides an inline function to reduse the function call overhead.

Inline function may increase efficiency of program when it is small.

-> Syntax:

```
inline return type function name(para-
                                 meters
{
   Function code
}
```

Inline function may not be useful for many embedded systems.

-> Example:

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

inline float mult(float x, float y)
{
  return(x*y);
}

inline float div(float r, float s)
{
  return (r/s);
}

int main()
{
  float a = 2;
  float b = 2;
  cout << mult(a, b) << endl;
  cout << div(a, b) << endl;
  return 0;
}
```

Page No.
Date : / /

8 Explain following with suitable example.

(1) This pointer

Every object in c++ has access to its own address through an important pointer called This pointer.

The This pointer is an implicit parameter to all member function.

Whenever a member function is called, it is automatically passed an implicit arguments.

This pointer is not available in static member function as static member function can be called without any object.

this is a keyword that refers to the current instance of the class.

-> Keyword: this

-> Syntax: this -> ~~Variable~~ data member;

-> Example:

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

class demo
{
  Private:
      int n;
      char ch;

  public:
      void setvalue(int n, char ch)
      {
        this -> num = num;
        this -> ch = ch;
      }

      void displayvalue()
      {
        cout << num << endl;
        cout << ch;
      }
```

```
};

int main()
{

demo ob;

ob.setvalue(1,'B');
ob.displayvalue();

return 0;

}
```

(2) Pointer to Object:

The object pointers are declared by placing in front of a object pointer's name.

You can access an object using a pointer to object.

When a pointer is incremented or decremented, it is increased or decrased in such a way

Pointer to object provide direct access to memory

Pointer to object are process data very fast

Pointer to object help in building complex data structures.

→ Syntax:

class name * Object pointer name

→ Example:

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

class Number
{
    int n;
public:
    void setvalue(int a)
    {
        n = a;
    }
```

```
    void shownum()
    {
      cout << num << endl;
    }
};

int main()
{
  Number ob, *p;

  ob.setvalue(2);
  ob.showvalue();

  p = &ob;
  p -> showvalue();

  return 0;

}
```

17 Illustrate with example working of endl, st setw manipulator.

Manipulator are helping function that can modify the input or output stream.

Manipulator does not mean that we change the value of variable.

=> Endl manipulator

Endl manipulator is type of manipulator with are exist without argument.

endl is defined in outstream.

endl is used to enter a new line and after entering a new line it flushes the outstream.

Working of manipulator is similar to the working of ('\n') in C++.

endl does not occupy any memory.

-> example:

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

int main()

{ ch name; int roll number;
  cout << "Name" << endl;
  cin >> name;

  cout << "roll number" << endl;
  cin >> roll number;

  return 0;

}
```

=> setw manipulator:

setw is a type of manipulator which has argument.

setw is used to set the field width in output operations.

setw is a manipulator of iomanip library.

setw does not returns ~~anting~~ anythings. It only acts as stream manipulator.

-> example:

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

  int main()
{
   int n=50;

   cout << "setting the width using
            setw to 5" <<endl;

   cout <<setw(5);

   cout <<n <<endl;

   return 0;

}
```

SMVS
Page No.
Date : / /

7 Explain the use of Destructor.

This are the main use of Destructor

1 Destructor is use for destroy constructor's work.

2 Destructor is use for destroy constructor's memory allocation.

3 Use of destructor, use less memory in program.

4 Destructor is never return any type of argument.

5 Destructor is never take any type of argument.

6 For declaration of destructor there no need any return type.

7 Destructor is reduce space of program.

19 Define class and Object.

-> Class: Class is a user-defined data type.

Class in c++ is the building block that leads to OOP.

When class is created, there is no memory allocated to the class.

-> Object: Object is an instance of a class.

When we created object, then memory is allocated to the class.

Syntax : classname objectname;

-> Write syntax for accessing a data member and member function in class.

-> Syntax:

```
class class name;
{
  data type variable decalration;
  Access Sepesifier:
    return type function name (argument);

    return type function name ();
    {
      // Function defination.
    }
};
```

-> How to define and access member Function inside the class with example.

SMVS
Page No.
Date : / /

A member function and data member of a class can be also defined inside the class.

When a member function is defined inside the class, the class name and scope resolution operator can not use.

The member functions defined inside a class definition are by default inline function.

Function define in class is ~~at~~ automatically treated as an inline function.

-> ~~em~~example:

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

  class (~~demo~~) student
  {
    Public:
        string name;
```

```
    void display()
    {
      cout<<"Khushi"<<endl;
    }
};

int main()
{
    student ob;
    ob.display();
    return 0;
}
```


20 Explain Operator Overloading.

Operator overloading means, we can use operator with the different argument in program.

Operator overloading allows a programmer to define new types from the built-in types.

Operator overloading should be used to perform the same name operator using different argument.

operator overloading should be either member function or friend function.

For operator overloading, at least one of the operands must be a user defined class object.

-> There are two types of Operator.
1) Binary Operator
2) Unary Operator

1 Binary Operator:

Friend function requires two argument for binary operator.

Member function requires one argument for binary operator.

2 Unary Operator:

Friend function requires one argument for unary operator.

Member function requires zero argument for unary operator.

-> example:

```
class demo
{
  private:
     int x;
  public:
     void getdata (int x)
     void display
     void operator -();
};

void demo :: getdata (int a)
{
    x = a;
}
```

```
void demo :: display ()
{
  cout <<     << endl;
}

void demo :: operator - ();
{
    x = - x;
}

int main ()
{
  demo c;
  c.getdata (1);
  cout << " c " << endl;

   c.display ();

   return 0;

  }
```

→ Syntax :

return type operator op(argument)
{
//Function body;
}

21 Explain Merits and Demerits of Inline function.

-> Merits of Inline function.

1 Inline function reduse the code of program.

2 Inline function may increase efficiency of program.

3 Inline function written only one single line.

4 It does not require function calling overhead.

5 It also save overhead of variable push on the stack while

function calling.

6 It also save overhead of return call from a function.

7 It increases locality of reference by utilizing instruction cache.

-> Demerits of Inline Function:-

1 It is increase size of the binary executable file.

2 Too much inling can also reduse your instruction cache hit rate.

3 Inline function is also increase size of header file.

4 It is not useful for embeded system.

5 Any change in the function code would need you to recompile the program.

6 The added variable from the inlined function consumes additional registers.

7 Use for more Inline function might cause thrashing.

25 Explain Access Specifier : Public, Protected and Private in Detail.

=> There are three type of Inheritance

1) Public Inheritance
2) Private Inheritance
3) Protected Inheritance.

1 Public :

Public function members are accessible from outside the class.

Public in derived class can be accessed directly by member, Friend function.

The Public data member is inheritate in derived the class.

It does not provide any security for its data member.

Public data member is accessible from the ~~th~~ whole class.

2 Private :

~~Pur~~ Private members are accessible from inside the class.

Private data members does not accessible from outside the class.

Private data members provides ~~less~~ more security ~~o~~ for its data members.

Private data member is not inheritate in derived class for any range.

SMVS
Page No.
Date : / /

3 Protected :

Protected data member are accessible in class.

Protected data member provides less security for data member.

Protected data member is inheritate in derived the class.

→ Syntax -

public :

private :

This are the Access Specifier in OOP.

14 Describe memory management operators in C++.

In C++, there two operatores are use for memory management.

1) New Operator
2) Delete Operator.

1 New Operator:

A new operator is used to create the new object in OOP.

When we use new operator for create object, then the object will exist until we use the delete operatoor.

The new operator is an operator which denotes a request for memory allocation on the heap.

The new operator calls the special function operator new.

-> Syntax:

pointer variable = new data type

-> Ex. int * p;
p = new int

2 Delete Operator:

A Delete operator is used to delete object in OOP.

When memory is no longer required, than it needs to deallocated memory then delete Operator is use.

The delete operator call the special function operator delete.

Delete Operator is use for deallocated memory in the OOP.

SMVS
Page No.
Date : / /

-> Syntax:

delete pointer variable;

-> ex: int * ptr1
delete ptr1;