Name - Sumit Kumar Giri

Uni. Roll No. - 2000213

Course - B.Tech (CSE)

Sem. - 3rd

Subject - Object oriented programming in C++

Subject Code - (ACCS-16302)

Assignment - 2nd.

Section-A

1.

(i) Differentiate between friend function and general member function of the class.

Ans:- There are many differentiate between friend function and general member function of the class.

Friend function	General member function
It can be declared in any number of classes using the keyword friend.	It can be declared only in the private, public or protected scope of a particular class.
This function has access to all private and protected members of classes.	This function has access to private and protected members of the same class.
one can call the friend function in the main function without any need to object.	one has to create an object of the same class to call the member function of the class.
The unary operator takes at least one explicit parameter.	The unary operator does not take any explicit parameter.

(ii) Write down the various limitations of inheritance.

Ans:- There are some various limitation of inheritance.

(a) Inheritance promotes reusability. when a class inherits or derives another class, it can access all the functionality of Inherited class.

(b) Reusability enhanced reliability.

(c) Inheritance makes the sub classes follow a standard interface.

(d) Inheritance facilitates creation of class libraries.

(iii) Is it always necessary to create objects from class

Ans:-

(iv) write down the syntax of pure virtual function. why do we require pure virtual function?

Soln:- The synatax of pure virtual function.

virtual <func_type><func_name>() = 0;

(v). Differentiate between early binding and late binding.

Ans:- There are many differentiate between early binding and late binding.

Early binding	late binding
It is a binding that happens at compile time.	It is binding that happens at run time.
Actual object is not used for binding.	Actual object is used for binding.
It is also called ~~early~~ static binding because binding happens during compilation.	It is also called Dynamic binding because binding happens at run time.
Method overloading is the best example of static binding.	Methode overriding is the best example of dynamic binding.
private, static and final methods show static binding.	Other than private, static and final methods show dynamic binding.

(vi) write down the characteristics of abstract class.

Ans:- The following are the characteristics of an abstract class.

(a) you cannot instantiate an abstract class directly. This implies that you cannot create an object of the abstract class; it must be inherited.

(b) you can have abstract as well as non-abstract members in an abstract class.

(c) you must declare at least one abstract method in the abstract class.

(d) An abstract class is always public.

Section-B

Q2. Explain the Concept of virtual destructor by using Suitable example.

Ans:- A virtual destructor is used to free up the memory space allocated by the derived class object or instance while deleting instances of the derived class using a base class pointer object. A base or parent class destructor use the virtual keyword that ensure both base class and the derived class destructor will be called at run time, but it called the derived class first and then base class to release the space occupied by both destructors.

Ex:-

```
#include<iostream.h>
#include<conio.h>
 class A
{
 public:
  A()
{
 Cout<<"Constructor A";
}
 {
  Cout<<"Destructor A";
 }};
 Class B : Public A
 {
  int *a;
  Public:
  ~B()
```

```
{
cout<<" Constructor B allocating 10 bytes of memory \n";
delete a=new int[5];
}
B()
{
cout<<"Destructor B frees 10 bytes of memory \n";
delete[]a;
}
};
int main()
{
clrscr();
B *ptr;
ptr= new B;
delete ptr;
return 0;
}
```

OUTPUT :-

```
Constructor A
Constructor A allocating 10 bytes of memory.
Destructor B.
```

Q3. write a program to overload binary operator.

Soln:-

```
# include< iostream.h>
# include< conio.h >
  Class A
  {
    int a;
   public
   void set_a();
   void get_a();
   A operator+(A);
};
  void A:: set_a()
  {
   a=10;
   }
  void A ::get_a()
   {
    Cout<<a<<"\n";
   }
A A :: operator+(A ob)
{
   A temp;
temp.a =a=a+ob.a;
return temp;
}
```

```
int main()
{
A ob1, ob2;
ob1. set_a();
ob2. set-a();
A ob3= ob1+ob2;
Cout << "The value of a after Calling operator overloading
                    function + is : ";
ob3. get_a();
}
```

OUTPUT:-

The value of a after calling operator overloading function is : 20

Q4. Explain Constructor and its types by using suitable programming examples.

Ans:- A constructor is a special member function having the same name as that of its class which is used to initialize some valid and allocate resource. It is excuted automatically whenever an object of a class is created. Constructor must be destructed by destructor.

There are three types of Constructors.

(i) Default Constructor.

(ii) Parameterized constructor.

(iii) Copy Constructor.

(i) Default Constructor :-

A Constructor which has no argument is known as defaut Constructor. It is Invoked at the time of creating object.

Example :-

```
# include< iostream.h>
# include< conio.h>
  class Employee
  {
    Public:
     Employee()
     {
      cout<<" Default Constructor Invoked "<<endl;
  }
  };
  int main()
  {
    Employee e1;
    Employee e2;
    return 0;
  }
```

OUTPUT

```
Default Constructor Invoked.
Default Constructor Invoked.
```

(ii) parameteriz

A construc
parameter
differ

Exampl

(ii) Parameterized Constructor :

A constructor which has parameterized is called parameterized constructor. It is used to proride different values to distinct objects.

Example :-

```
# include< iostream.h>
# include< conio.h>
  Class Employee
 {  public :
    int Id;
    String name;
    float salary;
   Employee (int i, string n, float s)
   {
     Id = i;
    name = n;
    Salary = s;
  }  void display ()
   {
    Cout<< Id <<" "<<name<<" "<< Salary <<endl;
   }
   };
   int main (void)
   {
    Employee e1 = Employee (101,"Sanoo", 890000);
```

```
    Employee e2 = Employee (102, Nakul, 590000);
    e1. display ();
    e2. display ();
    return 0;
}
```

OUTPUT :-

```
101 Sonoo 890000
102 Nakul 590000
```

(iii) Copy Constructor :-

A copy constructor is a member function which initializes an object using another object of the same class.

Example :-

```
# include < iostream.h >
# include < conio.h >
Class A
{
 Public :
 int x;
 A (int a)
 {
  x = a;
 }
 A (A & i)
 {
  x = i.x ;
 }
};
```

```
int main()
{
A.a1(20);
A.a2(a1);
Cout<<
return
}
```

590000);

```
int main()
{
  A.a1(20);
  A.a2(a1);
  Cout<< a2.x;
   return 0;
   }
```

OUTPUT :-

20.