Assignment - 2

(1)
Ans: (d) Stack

(2)
Ans: (C) Compile error in line "Derived * dp = newbase;"

(3)
Ans: (a) inaccessible

(4)
Ans: (a) The number of times destructor is called depends on number of objects created.

(5)
Ans: (a) True

Short Answer Question :-

(1)
Ans: The New Operator is an Operator which denotes a request for Memory allocation on the heap. if Sufficient Memory is available, New Operator initializes the Memory & returns the address of the newly allocated and initialized Memory to the pointer Variable.

for example, we can see that the Syntax for using the New Operator is

⇒ Pointer Variable = new data type.

Delete operator : Once we no longer need to use a variable that we have declared dynamically, we can deallocate the Memory occupied by the Variable.

For this the delete operator is used. it returns the Memory to the Operating system. This is known as Memory allocation.

⇒ The Syntax for this Operator is

delete pointer Variable ;

Code :-

```
C++ Dynamic Memory Allocation
#include < iostream>
Using namespace std;
int main () {
    // declare an int pointer
    int * pointInt;
    // declare a float pointer
    float * pointFloat;
```

```
// dynamically allocate memory
pointInt = new int;
pointFloat = new float;

// assigning value to the memory
*pointInt = 45;
*pointFloat = 45.45f;

cout << *pointInt << endl;
cout << *pointFloat << endl;

// deallocate the Memory
delete pointInt;
delete pointFloat;

return 0;
}
```

---

(2)

Ans :- A constructor is a special type of function with no return type. Name of constructor should be same as the name of the class. We define a Method inside the class and constructor is also defined inside a class. A constructor is called automatically when we create an object of a class.

⇒ Constructors initialize the New object i.e they set up the startup property values for the object. They might also do other things necessary to make the object usable.

You can distinguish constructors from other methods of a class because constructors always have the same name as the class.

Different types of Constructors :-

→ Default Constructor
→ Parametrized Constructor.
→ Copy Constructor.

1, Default Constructor :-

```
# include <bits/stdc++.h>
    using namespace std;
Class cube
{
public :
 int side ;
 cube ()
 {
   side = 10;
 }
};
int main ()
}
  Cube c;
  cout << c.side ;
}
```

2, Parameterised Constructors :-

```
# include <bits/stdc++.h>
using namespace std ;
```

```
class cube
{
  public :
  int side ;
  cube (int x)
  {
    side = x ;
  }
};
int main ()
{ Cube c1(10) ;
  Cube c2(20) ;
  Cube c3(30) ;
  Cout << c1. side ;
  cout << c2. side ;
  cout << c3. side ;
}
```

3) Copy constructor :- Copy constructors is a type of constructor which is used to create a copy of an already existing object of a class type. it is usually of the form X (X &), where X is the class name. The compiler provides a default copy constructor to all the classes.

Syntax of copy Constructor

```
Classname ( const class name & object name)
{
  .....
}
```

(3)

Ans:

Difference between procedural programming and Object Oriented programming :-

Procedural Oriented Programming	Object oriented programming.
1, In procedural programming, program is divided into Small parts called functions.	1, In object oriented programming program is divided into Small parts called objects.
2, procedural progaming follows top down approach.	it follows bottom up approach.
3, There is no access Specifier in procedural programming	This have access specifiers like private, public, protected etc,..
Adding new data & function is not easy.	Adding new data & function is easy.
does not have any proper way for hiding data so it's "less Secure"	provides data hiding so it's more Secure
Over loading is not possible	Over loading is possible.

function is more important than data	data is more important than function.
Based on unreal world	Based on real world.
Examples : C, Forfain, Pascal, Basic etc	Examples : C++, Java, Python, C#, etc.

long Answers Questions :

(1)

Ans: Polymorphism in C++ is a feature of oops that allows the object to behave differently in different conditions. in C++ we have two types of polymorphism.

→ Compile time polymorphism (static/early binding)

→ Runtime poly morphism. (dynamic / late binding)

↓ Compile time polymorphism :

Function overloading and operator overloading are perfect example of Compile time polymorphism.

Example : Here, we have two functions with same name but different number of arguments. Based on how many parameters we pass during function call determines which function is to be called, that is why

It is Considered as an example of polymorph
because in different conditions the output is differe

```
#include <iostream>
Using namespace std;
Class Add {
Public :
    int Sum (int num1 , int num2) {
        return num1 + num2 ;
    }
    int Sum (int num1 , int num2, int num3) {
        return num1+ num2+num3 ;
    }
};
int main () }{
    Add obj ;
    // This will call the first function
    cout << "Output : " << obj. Sum (10, 20) << end 1
    // This will call the second function
    cout << "Output : " << obj .Sum (11, 22, 33) ;
    return 0 ;
}
```

3, Runtime polymorphism :-

Function Overriding is an Example of Runtime polymorphism.

Function Overriding :- When child class declares a

Method, which a already present in the parent class then this is called function overriding, here child class overrides the parent class.

in case of function overriding we have two defints of the same functions, one is parent class and one in child class. the call to the function is determined at runtime to decide which def of the function is to be called, thats the reason it is called Runtime polymorphism.

Example :-

```
# include <iostream>
using namespace std;
class A {
public :
    void disp () {
        cout << "Super class function" << endl ;
    }
};
class B : public A {
public :
    void disp () {
        cout << " sub class function" ;
} };
int main () {
    // parent class object
    A obj ;
```

```
obj.disp();
// child class object
B obj2
obj2.disp();
return 0;
}
```

Ans :-

© Solution :-

```
# include < bits/stdc++.h >
Using namespace std ;
# include <iostream>
# include < string >
Using namespace stg ;

class member {

 char name [20], address [40] ;
  double number ;
  int age ;

  Public :
  int salary ;
  void input ()
  { cout << endl ;
   cout << " Name : " << endl ;
   cin . getline (name, 20) ;
   cout << " Age : " << endl ;
   cin >> age ;
   cout << " phone Number : " << endl ;
   cin >> number ;
   cout << " Address : " << endl ;
   cin . getline (address , 40) ;
   cout << " Salary : " << endl ;
   cin >> salary
```

```
Void display ()
  { Cout << endl ;
   Cout << "Name : "<<name<<endl;
   Cout <<" Age : " << age << endl ;
   Cout << "phone Number : " <<number << endl ;
   Cout << " Address : " << address <<endl ;
   Cout <<" Salary : " << salary<<endl;
}
};
class employee : public member {
  char specialization [20], department [20] ;
  public :
  void input ()
  {
  Cout << "\n\t Enter Employee Details \t \n";
  member :: input () ;
  Cout <<" Specialization : "<<endl;
  Cin. getline (specialisation , 20) ;
  Cout <<" Department : "<< endl;
  Cin.getline ( department ,20) ;
  }
  Void display ()
  {
  Cout <<"\n\t Displaying Employee Details \t
  \n";
  member :: display () ;
  Cout <<" Specialization : " << specialisation << endl ;
  Cout <<" Department : "<< department << endl;
  }
```

```
Void printsalary ()
  {
Cout << "\n Salary of the member is :"
<< Salary << endl;
  }
};

Class manager : public member {
   char Specialization [20], department [20];
   Public :
   Void input ()
  {
  Cout <<"\n\t Enter Manger Details \t\n";
  member :: input ();
  Cout << " Specialization :" <<endl;
   Cin.getline (specialization, 20);
   Cout << " Department :" <<endl ;
   cin. getline (department, 20) ;
  }
  Void display ()
  {
  cout <<"\n\t Displaying Manager Details \t\n";
  member :: display ();
  Cout << " Specialization : " << specialization << endl;
  cout << " Department : " << department << endl ;

}}
  Void print salary ()
   {
```

```
    cout << "\n Salary of the member is :"
<<salary <<endl;
    }
};
 int main ()
 {
 employee e;
 manager m;
 e.input ();
 m. input ();
 e.display();
 e. printSalary();
 m. display();
 m. print Salary();
}
```