Assignment - 2

Roshan kumar

* Choose the correct option:-

1d. Stacks

2c. Compiler error in line "Derived *dp = newBaric;"

3.a. Inaccessible

4.a. The number of times destructor is called depends on number of object created

5.a. True

* Short answer type questions

1. Syntax to use new operator: To allocate memory of any data type, the syntax is

pointer_variable = new datatype;

delete operator
Since it is programmers responsibility to deallocate dynamically allocated memory programmers are provided delete operator

Syntax

// Release memory pointed by pointer-variable
delete pointer_Variable;

Example

```
# include <iostream>
using namespace std;
int main()
{
  // pointer intialization to null
     int* p = Null
  // Request memory for the variable.
  // using new operator
     p= new (nothrow) int;
       if (!p)
cout <<" allocation of memory failed \n";
  else
  }
  // store value at allocated address
     *p = 29;
cout <<" Value of .p" <<*p<< endl;
   }
// Request block of memory
// using new operator
    float *r = new float (75.25);
```

```
cout<<"value of r;" <<*r<<endl;
    // Request block of memory q of 25120
int n=5;
int *q = new(nothrow) int [n];
  if (:q)
cout<<"allocation of memory feiled \n";
  else
{
  for(int i=0; i<n<i++)
    q[i] = i+1;
cout<<"Value store in block memory ';
    for (int i=0; i<n; i++)
cout<<q[i]<<" ";
}
// Free the allocated memory
    delete P;
    delete r;
// Free the block of allocated memory
    delete [] ];
    return 0;
}
```

(2.) A constructor is a ~~memory~~ member function of a class which intiidlizes objets of a class. In c++ constructor is automaticaelly called (when object (instemce of class) Creete. It is speial member function of the class

The main purpose of the class construction in c++ programming in to construct an object of the class.

Default constructor is the constructor which doesn't take any argument : It has no parameter.

For Example

```
class cube
{
  Public
  int side;
  Cube()
  {
    side = 10;
  }
}; int main()
{
  cube c;
  cout << c.side;
}
```

Parameterized Constructor

These are the constructor with parameter using the constructor you can provide different value to data members of different object by passing the appro. value as argument

(3)

Procedural Oriented Programming	Object oriented Programming
* In procedural programming program is divided into small parts called function	* In object oriented programming program is divided into small parts called objects.
* procedural programming follows top down approach	* object oriented progeamming follows botton up approach.
* There is no access specifiers in procedural programming	* object oriented programming have access specifiers like private, public, protected etc.
* Adding new data and function is not easy	* Adding new data and function is easy
* Procedural programming doesn't have any proper way for hiding data	* Oprated oriented programming provides data hiding so, it is more secure.
* Procedural programing is based on unreal world	* object oriented progeamming is based on real world.
* Ex- FORTRAN, Pascal, Basic etc	* Ex:- C++, Java, Python, etc.

Long Answer type question :-

(A.) Type of polymorphism

C++ Support Two Type of polymorphism :-

→ Compile time polymorphism - You invoke the overloaded function by matching the nubes and type of argument. The information is present during compile time. This means the C++ compiler will select the right function at compile time.

Compile time polymorphism is achieved through function overloading and operator overloading

Function Overloading

Function overloading occurs when we have many function with similar names but different arguments. The arguments may differ in terms of number or type.

Exaple

```
# include <iostream>
using namspace std;
void test (int i)
{
```

```
}
cout<<" The int i is "<<i<<endl;
}
void test (double f)
{
cout <<" The float is "<<f<<ew;
}
void tst (char cout *ch)
{
cout << "The char *is "<<ch<<endl;
}
int main
{
tst (5);
tess (5.5)
test ("Five");
return 0;
```

Operator Overloading

In operator overloading we define a new meaning for a c++ operator. It also change how the operators works. for exaple we can define the + operator to concatut two string we know it as the addition operator for adding numerical value after our defination when placed beween integers it will add them when placed between strings it will concatenate them.

Exauple

```
#include <iostream>
using namespace std;
class complex Num
{
private:
  int real, over;
public;
    complex (int rl=0 ; int ov=0)
{
   real =rl;
   over = ov;
}
complex Num operator + (complex Nw.cout &obj)
{
 complex Num result;
 result.real = real + obj.Real ;
 result.over = over + obj.ove;
 return result;
 }
   void print ()
 }
.cout << real << "i" << over << endl;
}
};
```

P.T.O

{
complex num c1(10,2), c2(3,7);
complex num c3 = c1 + c2;
c3.print();
}

Runtime Polymorphism -
This happens when objects method is invoked called during runtime rather than during compile time. runtime Runtime polymorphism is achieved through fuction overriding this fuction to be called invoked is established during runtime.

Function Overriding
Function overriding occurs when a function of the base class is given a new definition in a derived class. At that time, we call the base function as overriding.

Example:-
#include <iostream>
using namespace std;
class normal
{
cout

```
{
  Public :
  void eat()
{
  Cout <<"mammals eat ....";
}
};
class cow: public mammals {
    Public :
      void eat()
{
  cout<<"cows eat grass. ....." }
  };
  int main(void){
     cow c = cow();
     c.eat ();
     return 0;
  }
```

Virtual Function :—

A virtual function is another way of implementing run-time polymorphism in c++. It is a special fution defined in a base class and redefined in the derived class To declare a virtual keyword. They keyword should precede the declaration of the fution in the base class

If virtual function class is inherited the virtual class redefine the virtual fuction to suit its nuis. for exaple.

```
#include<iostream>
using namspce std;
class classA
{
public:
virtual void show()
{
cout<<"The show fuction in
base class invoked...."<<endl;
}
};
class classB: public classA
{
public:
void show()
{
t<<"The show fuction in derive
class invoked ....";
}
};
int main();
{
class A* a;
class B b;
a = &b;
a -> show();
}
```

(B)

```
// C++ program to sort an array
// with 0, 1, and 2 in a single pass
#include <bits/stdc++.h>
using namespace std;
// Function to sort the input array
// the array is assumed
// to have values in {0, 1, 2}

void sort12(int a[], int arr_size)
{
    int lo = 0;
    int hi = arr_size - 1;
    int mid = 0;
    // Iterate till all the elements
    // are sorted
    while (mid <= hi)
    {
        switch (a[mid])
        {
        // If the element is 0
        case 0:
            swap(a[lo++], a[mid++]);
            break;
        // If the element is 1
        case 1:
            mid++;
            break;
```

```
If the elemet is 2
  cnt 2;
  Swap (a[mid] a[h--]);
  break;
}
}
}
// Fuction to print array arr[]
  void print array (int arr[], int arr-si
}
// Ftarate and procate every elemet
  For (int i=0; i<arr-size; i++)
    Cout << arr[i] << " ";
}
// Drive Code
  int main()
}
int arr[]={1,1,2,2,0,0,2,1,2};
  int n= size of (arr) / size of (arr[0])
    sort 012 (arr, n);
  Cout << "array after reprogiftion";
  print array (arr, n);
  return 0;
3
```

(c)

```
class member
{
String name;
int age;
String number;
String addres;
int salary;

Public void print salary ()
}
System out point n (salary );
}
  }
class employee extends member
}
 String specialization
}
class manager extends member
}
 String department
}
class Ame
}
 Public static void main (String [] argm)
}
```

Employee e = new employee ();
e.name = "xyz"
e.age = 23;
e.mubes = "9860xxxx000";
e.address = "xyzxyz";
e.salary = 1231;
e.specialization = "xyzxyz";

Manager m = new manager
m.name = "xyz"
m.age = 23;
m.number = "9860xxxx000";
m.address = "xyzxyz";
m.salary = 1281;
m.specialization = "xyzxyz";
}