DATE : / 20
PAGE No

Assignment - 2

Ankur

M.C.Qs :-

Stack

Compiler error in line "Derived *dp = new Base";

Inaccessible

The number of times destructor is called depends on number of objects created.

True

Short Answer type Questions

new :- To allocate memory of any data type

⇒ Pointer - Variable = new data-type;

delete :- used to deallocate memory

⇒ delete pointer - variable

DATE
PAGE No

Example:-

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int *P = NULL;
    P = new (nothrow) int;
    if (!P)
cout << "allocation of the memory failed \n";
else
    *P = 89;
cout << "Value of P: " << *P << endl;
}
    float *r = new float (12.8);
    cout << "Value of r: " << *r << endl;
    int n=8;
    int *q = new (nothrow) int [n];
if (!q)
    cout << "allocation on the memory failed \n";
else {
        for (int i=0; i<n; i++)
    q[i] = i+1;
cout << "Value store in block of memory: ";
for (int i = 0; i<n; i++)
cout << q[i] << " "; }
delete P;
delete r;
delete []q;
return 0;
```

DATE : / 20
PAGE No

A constructor is a member function of a class which initializes objects of a class. In C++ constructor is automatically called object create. It is a special member function of class. The main purpose of class constructor in C++ programming is to construct an object of a class. In other words it is used to initialize all class data members.

Types :-

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

Syntax :- {Class name (parameter 1, parameter 2 ...)}

Example :-
```
#include <bits/stdc++.h>
using namespace std;
class Cube
{
Public
int side;
cube ()
{ side = 10;
}}
int main ()
```

DATE : / 20
PAGE No

```
{
  cube c;
cout << c.side;
  return 0; }
```

Parameterized constructors:- These are the constructors with parameter. using this, you can provide different values to data members of different objects, by passing the appropriate values or argument.

Ex:-
```
#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;
  return 0;
```

DATE : / / 20
PAGE No

3. Copy constructor :- It 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 class name. The compiler provides a default copy constructor to all the classes.

Syntax :- Class name (const classname & Objectname)
{ - - - }

ex :-

```
#include <bits/stdc++.h>
using namespace std;
class Sample copy constructor
{
  Private:
int x, y;
Public:
Sample copy constructor (int x1, int y1)
{
   x = x1;
   y = y1;
}
Sample copy constructor (const Sample copy & sam)
{
  x = sam.x;
}y = sam.y;
   Void display()
```

DATE : / 20
PAGE No

```
int main()
{
Sample copy constructor obj 1 (10, 15);
Sample copy constructor obj 2 = obj 1;
Cout << "normal constructor :";
obj 1. display ();
Cout << "copy constructor :";
obj 2. display ();
return 0;
}
```

Q3 -

Object oriented.	Procedural oriented.
1. In OOP program is divided into small parts called objects.	In procedural program divided into functions.
2. It follows bottom up approach.	It follows top-down approach
3. It an access specifier like public, private & protected etc.	There is no access specifier in procedural programming.
4. Adding new data & function is easy.	Adding new data & function is not easy.

DATE: / / 20
PAGE No

It provides data hiding so, it's more secure.	Adding one. It doesn't have any proper way.
It is based on real world.	It is based on unreal world.
Ex:- C++, Java, python	Ex:- C, Pascal etc →

Long Answer

Polymorphism:- The word polymorphism means having many forms. It means that a call to a member function will cause a different function to be executed depending on the type of object invokes the function.

(i) compile time (ii) Runtime.

Compile Time:- also known as static binding. Function overloading & operator overloading are perfect example.

Example:-

DATE : / 20
PAGE No

```
#include <bits/stdc++.h>
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;
cout<<"output :"<< obj.sum (10,20)<<end
cout <<"output :"<< obj.sum (11, 22, 33);
return 0;
}
```

Runtime :- Also known as dynamic binding

Example :-

```
#include <bits/stdc++.h>
using namespace std;
class A {
public:
void display (){
cout << "super class function" <<end;
}}
```

9.

DATE : / 20
PAGE No

```
class B : public A {
public:
void disp() {
cout << "subclass function";
}};
int main() {
A obj;
obj.disp();
B obj2;
obj2.disp();
return 0;
}
```

Code :-

```
#include <bits/stdc++.h>
using namespace std;
void o12(int a[], int arr_size)
{ int lo = 0;
int hi = arr_size - 1;
int mid = 0;
while (mid <= hi) {
switch (a[mid]) {
case 0:
swap (a[lo++], a[mid++]);
break;
case 1:
mid++;
break;
```

DATE : / 20
PAGE No

```
case 2
  swap( a[mid], a[hi--]);
  break;
}}}
void printArray( int arr[], int arr_size)
{
  for (int i=0; i<arr_size; i++)
    cout << arr[i] << " ";
}
int main()
{
  int arr[] = {1,1,2,2,0,0,2,1,2};
  int n = sizeof(arr)/sizeof(arr[0]);
  sort012(arr, n);
  cout << "array after segregation: ";
  printArray(arr, n);
  return 0;
}
```

3.

```
#include <bits/stdc++.h>
  using namespace std;
class member{
char name[20], address[50];
  int number;
  int age;
public:
  int salary
```

DATE : / 20
PAGE No

```
void input ()
{
cout << "Name : " << endl;
cin >> name;
cout << "Age : " << endl;
cin >> age;
cout << "phone number : " << endl;
cin >> number;
cout << "Address : " << endl;
cin >> address;
cout << "Salary: " << endl;
cin >> salary;
}
void display ()
{
cout << endl;
cout << "Name : " << name << endl;
cout << "Age " << age << endl;
cout << "phone number : " << number << endl;
cout << "Salary : " << salary << endl;
};
class employee : public member {
char specialization [30], department [20];
public:
void input ()
{
cout << "\n Enter employee details : ";
member :: input ();
```

DATE : / 20
PAGE No

```
cout << Specilgation : " << endl;
cin >> specilgation
cout << " Department : " << endl;
cin >> department;
}
void display ()
{
cout << "\n Displaying employee details \n
member :: display ();
cout << " specilgation: " << specilgation << e
cout << " department : " << department << e
}
void print salary ()
{
cout << "\n salary of the member is :
" << salary << endl }};
cin >> specilgation
cout << " Department : " << endl;
cin >> department;
}
void display ()
{
cout << \n displaying manager details \n"
member :: display ();
cout << " Specilgation : " << sperilgation << e
cout << " department : " << department << end
}
void Print Salary () {
```

DATE: / / 20
PAGE No

```
cout << "\n Salary of the member is:
" << Salary << endl << 33;
int main()
{
Employee e;
manager m;
e.input();
e.display();
e.printsalary();
m.display();
m.printsalary();
return 0;
}
```

Output:-

```
Enter Employee Details:
Name: John
Age: 21
Phone Number:- 0123456789
Address: Punjab
Sallary:- 1200000
Specilization: developer
Department: CSE
Enter Manager Details:
Name: Larib
Age: 28
Phone Number: 01237678
Address: BGP
Salary: 1280000
Specilization: Coder
```

DATE: / /
PAGE No.

Department : CSE

Displaying Employee Details:-
Name : John
Age : 21
Phone Number : 0123456789
Address : Punjab
Salary : 1200000
Specialization : developer
Department : CSE
Salary of the member is 1200000

Displaying Manager details

Name : Jarasb
Age : 28
Phone Number : 01234567890
Address : Bgp
Salary : 1280000
Specialization : coder
Department : CSE

Salary of the member is 1280000