Assignment-2

Choose the correct answers:-

① Stack.

② Compiler Error in line "Derived *dp = new Base;"

③ Inaccessible.

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

⑤ ~~False~~. True.

Short Answers

① "new" Keyword:- The new keyword denotes request for memory allocation on the Heap. If sufficient ~~mee~~memory is available, then memory is ~~allocated~~ initialized. This happens when we create an object of a class using the new keyword.

Example:-

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

int main() {
  int i, *p, n;
  cout << "Enter the size : ";
  cin >> n;
```

```
p = new int[n];

if (p==NULL) {
    cout<<"\n memory allocation failed"; exit(1);
}
cout<<"\n Enter the elements:"<<endl;
for (int i=0; i<n; i++) {
    cin>>*(p+i);
}
cout<<"Elements entered are :"<<endl;
for (int i=0; i<n, i++) {
    cout<<*(p+i)<<endl;
}
}
```

"delete" keyword :- The delete keyword is used to delete the memory allocated dynamically which is created by using the new keyword as the memory allocated cannot be deleted automatically. If the delete keyword is not used after new keyword, this may lead to stack overflow.

Example:-

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

int main() {
    int *arr = new int[10];

    delete [] arr;

    return 0;
}
```

② A constructor is a member function of a class which initializes objects of a class. Constructor is automatically called when an object is created. It's main purpose is to construct an object of a class. There are three types of constructors:

(i) Default Constructor:- Default constructor is the constructor which doesn't take any argument. It has no parameters.

Example:-

```
construct() {
    x = 100;
}
```

(ii) Parameterized Constructors:- Parameterized conductor is the conductor in which we can pass through arguments. We use parameters to initialize the object.

Example:-

```
Student (int x, int y) {
    roll_no = x;
    age = y;
}
```

This is a parameterized constructor and can be called later on.

(iii) Copy Constructor:- A copy constructor is a member fundion which uses an existing object of a class to initialize a new one.

Example:-

```
A (int a) {
    x = a;
}
A A &p
// Below is a Copy Construct
A (A &p) {

}
```

③ Differences between object oriented and procedural programming language are as follows:-

	Object oriented	Procedural
①	In object oriented programming, prog programs are divided into small parts called objects.	In procedural programming, programs are divided into small parts called functions.
②	It follows bottom-up approach.	It follows top-down approach.
③	It is more secure as it provides data hiding.	It is less secure as it does not have a proper way of hiding data.
④	Overloading is possible.	Overloading is not possible.
⑤	It is based on real word.	It is based on unreal world.

Long answers:

① Polymorphism is having same function or object behave differently in different situations. There are two types of polymorphism.

(i) Compile time polymorphism:- Compile time polymorphism happens when the compiler selects the appropriate function at compile time. It is achieved by function overloading and operator overloading which is also known as static binding.

Example:-

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

class Add {
public:
  int sum(int a, int b) {
      return a+b;
  }
  int sum(int a, int b, int c) {
      return a+b+c;
  }
```

```
};

int main(){
  Add obj;

  cout << "Output : " << obj.sum(10,20) << endl;
  cout << "Output : " << obj.sum(15,30,45) << endl;
  return 0;
}
```

(ii) Runtime Polymorphism :- Run time polymorphism is achieved when the object's method is invoked at runtime instead of compile time. This is achieved by method overriding which is also known as dynamic binding.

Example:-

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

class A {
public:
  void display() {
    cout <<"Super Class Function"<<endl;
  }
};
class B {
public:
  void display() {
    cout <<"Sub Class Function"<<endl;
  }
};
int main() {
  A obj;
    obj.display();
    B obj1;
    obj1.display();
    return 0;
}
```

② //Program to sort an array of 0,1,2

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

void printArr(int arr[], int n) {
    for (int i=0; i<n; i++)
        cout << arr[i] << " ";
}

void sortArr(int arr[], int n) {
    int i=0, count0=0, count1=0, count2=0;

    for (i=0; i<n; i++) {
        switch (arr[i]) {
        case 0:
            count0++;
            break;
        case 1:
            count1++;
            break;
        case 2:
            count2++;
            break;
        }
    }
```

```
    i = 0;
    while (count0 > 0) {
        arr[i++] = 0;
        int 0 = 1;  count0--;
    }
    while (count1 > 0) {
        arr[i++] = 1;
        count1--;
    }
    while (count2 > 0) {
        arr[i++] = 2;
        count2--;
    }
    printArr(arr, n);
}

int main() {
    int arr[] = {1,1,2,2,0,0,2,1,2};
    int n = sizeof(arr) / sizeof(int);

    sortArr(arr, n);
    return 0;
}
```

③
```
#include <bits/stdc++.h>
using namespace std;

class member {
  char name[20], address[50];
  double number;
  int age;

  public:
    int salary;
    void input() {
      cout <<"Name : " <<endl;
      ~~cin >> getline(name,20);~~ cin.getline(name,20);
      cout <<"Age : " <<endl;
      cin >> age;
      cout <<"Phone Number : " <<endl;
      cin >> number;
      cout <<"Address : " <<endl;
      ~~cin >> getline~~
      cin.getline(address,50);
      cout <<"Salary : " <<endl;
      cin >> salary;
    }
```

```
void display() {
    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 Enter Employee Details \n";
    member :: input();
    cout << "Specialization : " << endl;
    cin.getline(specialization, 20);
    cout << "Department : " << endl;
    cin.getline(department, 20);
  }
```

```
    void display() {
      cout << "\n Displaying Employee Details\n";
      member::display();
      cout << "Specialization : " << specialization << 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() {
      member::input();
      cout << "Specialization : " << endl;
      cin.getline(specialization, 20);
      cout << "Department : " << endl;
      cin.getline(department, 20);
    }
```

```
    void display(){
        cout<<"Specialization : "<<endl; specialization<<endl;
        cout<<"Department : "<<department<<endl;
    }
    void printSalary(){
        cout<<"\n Salary of the member is : "<<salary<<endl;
    }
};

int main(){
    employee e;
    manger m;
    e.input();
    m.input();
    e.display();
    m.display();
    e.printSalary();
    m.display();
    m.printSalary();
}
```