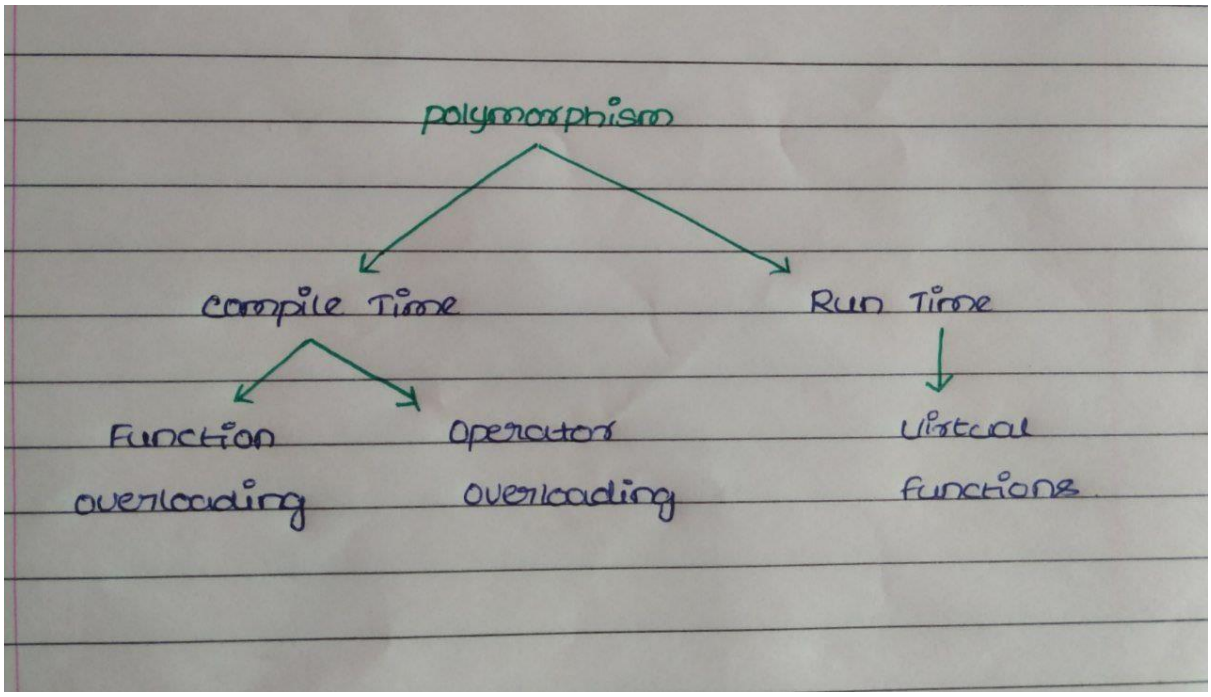
Long answer Type question.

A) Explain the type of polymorphism with code.

we can define polymorphism as the ability of a message to be displayed in more than one form. A real-life example of polymorphism, a person at the same time can have different characteristics. Like a man at the same time is a father, a husband, an employee. So the same person possess different behaviour in different situations. This is called polymorphism. Polymorphism is considered as one of the important features of Object-Oriented Programming.

In C++ polymorphism is mainly divided into two types:

- Compile time Polymorphism
- Runtime Polymorphism



1. Compile time polymorphism: This type of polymorphism is achieved by function overloading or operator overloading.

- **Function Overloading**: When there are multiple functions with same name but different parameters then these functions are said to be **overloaded**. Functions can be overloaded by **change in number of arguments** or/and **change in type of arguments**.
- **Operator Overloading:** C++ also provide option to overload operators. For example, we can make the operator ('+') for string class to concatenate two strings. We know that this is the addition operator whose task is to add two operands. So a single operator '+' when placed between integer operands, adds them and when placed between string operands, concatenates them.

2. Runtime polymorphism: This type of polymorphism is achieved by Function Overriding.

- Function overriding on the other hand occurs when a derived class has a definition for one of the member functions of the base class. That base function is said to be overridden.

```
// C++ program for function overloading
```

Polymorphism.cpp U

Assignment-02 > Polymorphism.cpp > main()

```cpp
#include <bits/stdc++.h>
using namespace std;
class Geeks
{
    public:
      // function with 1 int parameter
    void func(int x)
    {
        cout << "value of x is " << x << endl;
    }
     // function with same name but 1 double parameter
    void func(double x)
    {
        cout << "value of x is " << x << endl;
    }
      // function with same name and 2 int parameters
    void func(int x, int y)
    {
        cout << "value of x and y is " << x << ", " << y << endl;
    }
};
  int main() {

    Geeks obj1;
    obj1.func(7);
      // The second 'func' is called
    obj1.func(9.132);
    // The third 'func' is called
    obj1.func(85,64);
    return 0;
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\hp\Desktop\Assignments> cd "c:\Users\hp\Desktop\Assignments\Assignment-02\" ; if ($?)
phism }
value of x is 7
value of x is 9.132
value of x and y is 85, 64
PS C:\Users\hp\Desktop\Assignments\Assignment-02>
```

-
- `// CPP program to illustrate Operator Overloading`


```
// C++ program for function overriding
```

Polymorphism.cpp U Operator_overloading.cpp U Runtime_polymorphism.cpp U

Assignment-02 > Runtime_polymorphism.cpp > ...

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

class base
{
public:
    virtual void print ()
    { cout<< "print base class" <<endl; }

    void show ()
    { cout<< "show base class" <<endl; }
};

class derived:public base
{
public:
    void print () //print () is already virtual function in derived class, we co
    { cout<< "print derived class" <<endl; }

    void show ()
    { cout<< "show derived class" <<endl; }
};
//main function
int main()
{
    base *bptr;
    derived d;
    bptr = &d;
    //virtual function, binded at runtime (Runtime polymorphism)
    bptr->print();
    // Non-virtual function, binded at compile time
    bptr->show();
    return 0;
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
PS C:\Users\hp\Desktop\Assignments\Assignment-02> cd "c:\Users\hp\Desktop\Assignments\Assign
rphism } ; if ($?) { .\Runtime_polymorphism }
print derived class
show base class
PS C:\Users\hp\Desktop\Assignments\Assignment-02>
```

B) Write a program to sort an array of 0,1,2 in the best possible time and space complexity.

Given an array **A[]** consisting 0s, 1s and 2s. The task is to write a function that sorts the given array. The functions should put all 0s first, then all 1s and all 2s in last.

Example:

Input: {0, 1, 2, 0, 1, 2}

Output: {0, 0, 1, 1, 2, 2}

Input: {0, 1, 1, 0, 1, 2, 1, 2, 0, 0, 0, 1}

Output: {0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2}

Approach

1. a[1..Lo-1] zeroes (red)
2. a[Lo..Mid-1] ones (white)
3. a[Mid..Hi] unknown
4. a[Hi+1..N] twos (blue)
5. If the ith element is 0 then swap the element to the low range, thus shrinking the unknown range.
6. Similarly, if the element is 1 then keep it as it is but shrink the unknown range.
7. If the element is 2 then swap it with an element in high range

- **Complexity Analysis:**

Time Complexity: O(n).

Only one traversal of the array is needed.

Space Complexity: O(1).

No extra space is required.

Approach: Count the number of 0s, 1s and 2s in the given array. Then store all the 0s in the beginning followed by all the 1s then all the 2s.

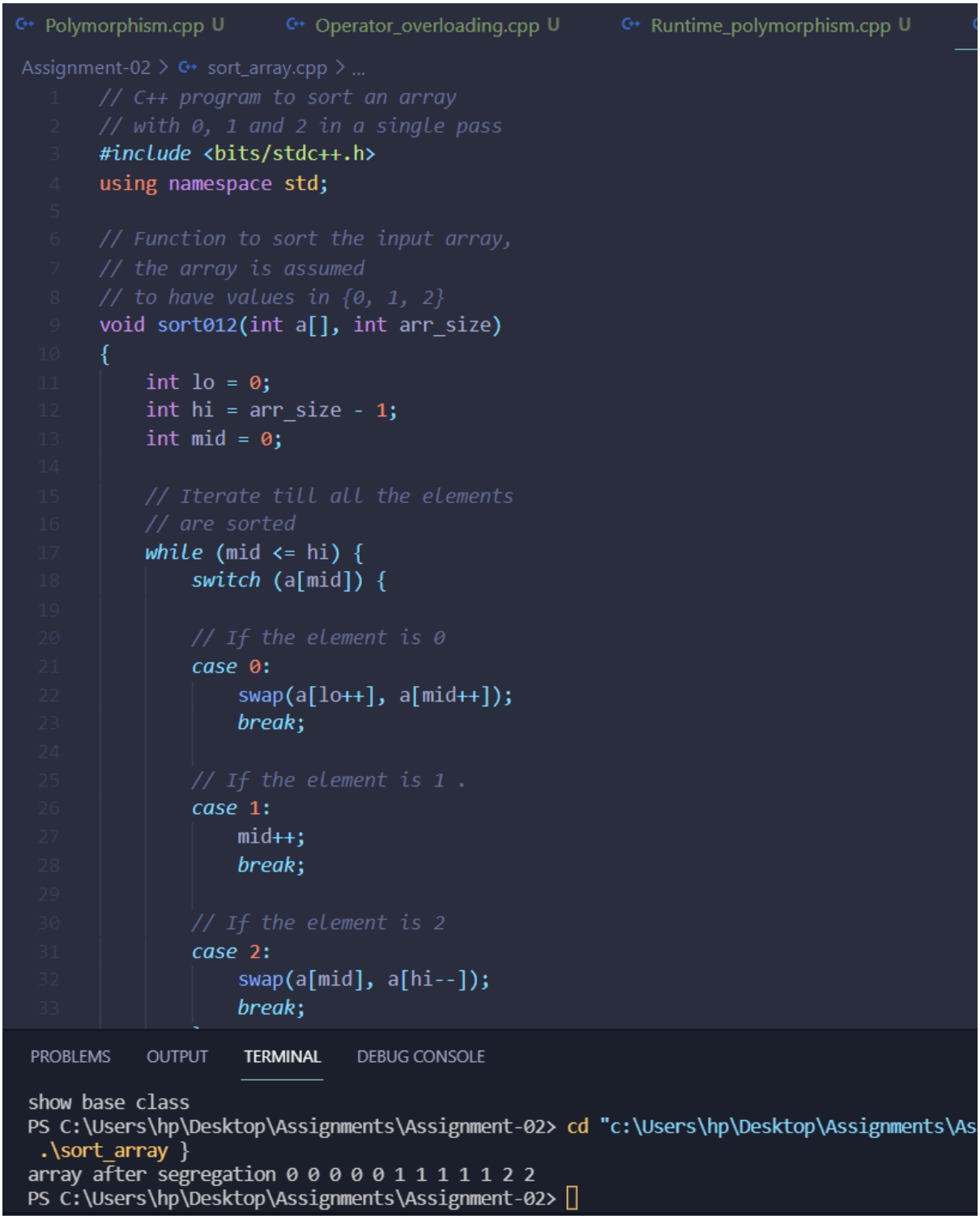
Algorithm:

1.Keep three counter c0 to count 0s, c1 to count 1s and c2 to count 2s

2.Traverse through the array and increase the count of c0 is the element is 0, increase the count of c1 is the element is 1 and increase the count of c2 is the element is 2

3.Now again traverse the array and replace first c0 elements with 0, next c1 elements with 1 and next c2 elements with 2.

```
// C++ program to sort an array
```



Create a class named 'Member' having the following members:
Data members
1 - Name
2 - Age
3 - Phone number

4 - Address
5 - Salary
It also has a method named 'printSalary' which prints the salary of the members.
Two classes 'Employee' and 'Manager' inherits the 'Member' class. The 'Employee' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign name, age, phone number, address and salary to an employee and a manager by making an object of both of these classes and print the same.

Long_03.cpp > member > input()

```cpp
#include <iostream>
#include <string>
using namespace std;

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:
```

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

Short answer type question

1.Explain about new and delete keywords with code

new operator

The new operator denotes a request for memory allocation on the Free Store. If sufficient memory is available, new operator initializes the memory and returns the address of the newly allocated and initialized memory to the pointer variable.

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

pointer-variable = new data-type;

Here, pointer-variable is the pointer of type data-type. Data-type could be any built-in data type including array or any user defined data types including structure and class.

Example:

// Pointer initialized with NULL

// Then request memory for the variable

int *p = NULL;

p = new int;

delete operator

Since it is programmer's responsibility to deallocate dynamically allocated memory, programmers are provided delete operator by C++ language.
Syntax:

// Release memory pointed by pointer-variable

delete pointer-variable;

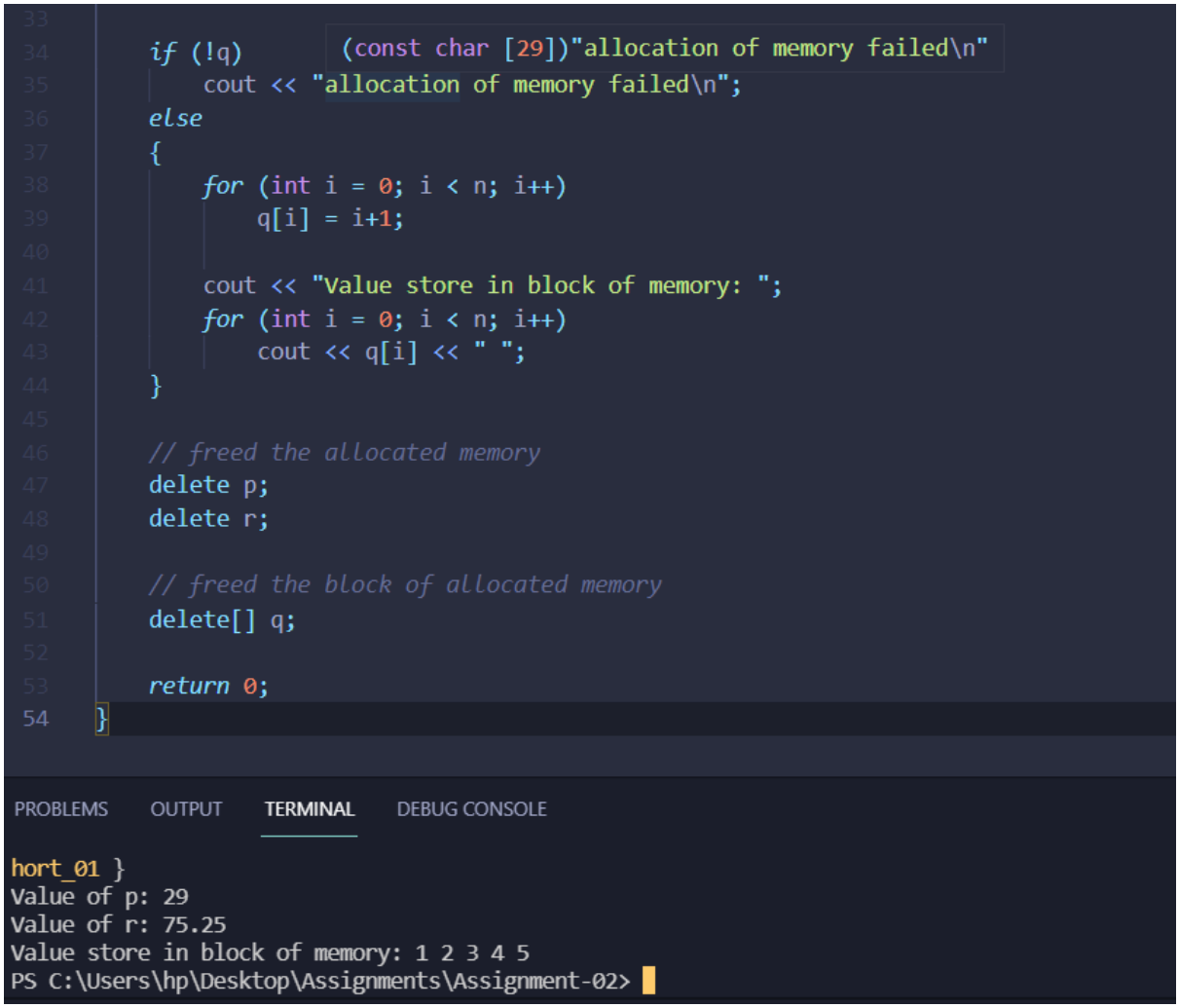
Here, pointer-variable is the pointer that points to the data object created by *new*.
Examples:

delete p;

delete q;

Assignment-02 > Short_01.cpp > main()

```cpp

// C++ program to illustrate dynamic allocation
// and deallocation of memory using new and delete
#include <iostream>
using namespace std;

int main ()
{
    // Pointer initialization 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 of size n
    int n = 5;
    int *q = new(nothrow) int[n];

    if (!q)
        cout << "allocation of memory failed\n";
    else
    {
        for (int i = 0; i < n; i++)
            q[i] = i+1;

```



```
    if (!q)        (const char [29])"allocation of memory failed\n"
        cout << "allocation of 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] << " ";
    }

    // freed the allocated memory
    delete p;
    delete r;

    // freed the block of allocated memory
    delete[] q;

    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

hort_01 }
Value of p: 29
Value of r: 75.25
Value store in block of memory: 1 2 3 4 5
PS C:\Users\hp\Desktop\Assignments\Assignment-02>
```

2.What are constructors? Why they are required? Explain different types of constructors with suitable example.

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. We can't call a constructor explicitly. Let us see the types of constructor.

Constructor Types

1. Default Constructor
2. Parameterized Constructor
3. Copy Constructor
4. Static Constructor
5. Private Constructor

Default Constructor

Default constructor does not take any parameter. C# compiler creates a default constructor if we do not define any default constructor inside the class, We can also define the default constructor by writing some codes. I defined a default constructor to assign properties of a class.

```
public  class Adminclass
    {
        string userId = string.Empty;
        string password = string.Empty;
       //Default constructor, having no any parameter
       public Adminclass()
       {
           userId = "shr";
           password = "Pass";
       }
    }
```

In the above example, I defined one class named as Adminclass and one default constructor with the same name as the name of the class. I used default constructor to assign the value to the private properties of the class. This constructor will automatically call when we create an object of this class.

How Default constructor is called.

```
Adminclass adminObject = new Adminclass();
```

Parameterized Constructor

Parameterized Constructor is created by the developer. This constructor takes at least one parameter. I have defined one Parameterized constructor below.

```
public  class Adminclass
    {
        string userId = string.Empty;
        string password = string.Empty;
       //Parameterrized constructor, having parameters
       public Adminclass(string username,string userPassword)
       {
           userId = username;
           password = userPassword;
       }
     }
```

In the above example, I defined one class named as Adminclass and one parameterized constructor with the same name as the name of the class. This constructor takes two parameters as you can see in the above example. I used a

parameterized constructor to assign values to the private properties of the class. This constructor will automatically call when we create an object of this class. Let us see how we can call this Parameterized constructor.

How Parameterized constructor will be called.

Adminclass adminObject = new Adminclass("Shreesh","mypassword");

3.Explain the difference b/w object oriented and procedural programming language in detail.

Procedural Programming:
Procedural Programming can be defined as a programming model which is derived from structured programming, based upon the concept of calling procedure. Procedures, also known as routines, subroutines or functions, simply consist of a series of computational steps to be carried out. During a program's execution, any given procedure might be called at any point, including by other procedures or itself.

Languages used in Procedural Programming:

FORTRAN, ALGOL, COBOL,

BASIC, Pascal and C.

Object Oriented Programming:
Object oriented programming can be defined as a programming model which is based upon the concept of objects. Objects contain data in the form of attributes and code in the form of methods. In object oriented programming, computer programs are designed using the concept of objects that interact with real world. Object oriented programming languages are various but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types.

Languages used in Object Oriented Programming:

Java, C++, C#, Python,

PHP, JavaScript, Ruby, Perl,

Objective-C, Dart, Swift, Scala.

Procedural Oriented Programming	Object Oriented Programming
In procedural programming, program is divided into small parts called ***functions***.	In object-oriented programming, program is divided into small parts called ***objects***.
Procedural programming follows ***top-down approach***.	Object oriented programming follows ***bottom-up approach***.

Procedural Oriented Programming	Object Oriented Programming
There is no access specifier 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 does not have any proper way for hiding data, so it is ***less secure***.	Object oriented programming provides data hiding so it is ***more secure***.
In procedural programming, overloading is not possible.	Overloading is possible in object-oriented programming.
In procedural programming, function is more important than data.	In object-oriented programming, data is more important than function.
Procedural programming is based on ***unreal world***.	Object oriented programming is based on ***real world***.
Examples: C, FORTRAN, Pascal, Basic etc.	Examples: C++, Java, Python, C# etc.

Choose the correct option

1. D
2. C
3. A
4. A
5. TRUE