CIE – I

Date :- 27/10/2021

Course/Program :- B. Tech.

Semester :- 3rd

Branch :- Computer Science & Engineering

Subject :- Object Oriented Programming using C++

Roll No. :- 2007100070 62

Total No. of pages :- 5

Zubayer Ahmed Zidhan Laskar

Q.1) ~~Sol~~

Ans:- The basic concepts of Object Oriented Programming are as follows :-

(i) Class :- A class is a blueprint to build a specific type of object. It determines how the object will behave and what properties it will have.

(ii) Object :- It is an instance of a class. It is the most basic element in Object Oriented Programming (OOP)

(iii) Polymorphism :- It is the ability of an operation to have many forms and do different things. This can be made possible through function Overloading or Operator Overloading.

(iv) Encapsulation :- It is the binding together of different types of data and operations under a single roof i.e. class.

(v) Abstraction :- It is the process of hiding all the implementation details and only displaying the essential details to the user.

(vi) Inheritence :- It is the process of a derived class inheriting the properties of the base class.

(vii) Message passing :- It is the ability of the different objects to communicate with each other by sending/receiving data.

//

~~③ Ans~~

3) (a) Ans :-

```
class INTEGER
{
    private:
        int   data;
    public:
        INTEGER (int objData)
        {
            data = objData;
        }

        INTEGER   operator+(INTEGER &obj2)
        {
            data = data + obj2.data;
            return  data;
        }
};
```

3) (b) Ans :-

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

class Operation
{
    ~~private:~~
        ~~int a, b;~~

    public:
        int a, b;
        Operation (int num1, int num2)
        {
            a = num1;
            b = num2;
        }
        virtual ~~void~~ int operate()  = 0;
};
```

Contd.

```
Class ~~add~~ add : public Operation
{
    public:
        ~~void~~ int Operate ()
        {
                return  a+b;
        }
};

~~class sub~~
Class sub : public Operation
{
    public:
        int operate()
        {
             return  a-b;
        }
};

int main ()
{
    add   addObj (5,2);
    sub   subObj (5,2);
    ~~addObj. operate~~
    cout << addObj. operate () << endl;
    cout <<  subObj. operate () << endl;

    return 0;
}
```

Expected Output:

```
7
3
```

3) c Ans: The difference between the two concepts is that in 3.a, operator overloading was used, whereas, in 3b, function overloading was used.

In operator overloading, we can make the operators of C++ behave in a certain way according to our need. Only a few C++ ~~to~~ operator cannot be overloaded.

In function overloading, we can have multiple functions performing different tasks with the only difference being in their parameters or return values or classes, and the compiler will know which function we want.

2) Ans:- Abstract Data Type (ADT) is a type (class) of objects whose behaviour is defined by a set of values and operations. The definition of an ADT only mentions what operations can be/are to be performed, but it does not explain how these operations are implemented.

Abstract Data Types help make things easier for the programmer/user who just needs to use the operations ~~a~~ defined in the ADT, and so doesn't need to care about how this was implemented or what algorithms were used for the implementation, and so ADT makes workflow faster as and ADT programmer can just create the ADTs and an user can just use them. ~~to~~

For example, let us assume we need to work with complex numbers frequently, but there is ~~not~~ complex type data prebuilt in C++, so, and ADT programmer could just create the complex data type, and give it the properties like real part, imaginary part, how to add/subtract/operate on complex numbers. And as an user, we can just create an "complex" type objects and use them as we need, without even thinking about the implementations. ~~For~~ us, complex becomes like int, float, etc., it just exists and we can use it.

—X—