Number of printed pages: 04

Semester II (B.Tech.) **Er. No……………….**

Academic Year: 2021-22

Jaypee University of Engineering & Technology, Guna

T-2 (Even Semester 2022)

18B11CI211 – OBJECT ORIENTED PROGRAMMIMG

Maximum Duration: 1 Hour 30 Minutes Maximum Marks: 25

Notes:

1. This question paper has ***five*** questions.
2. Write relevant answers only.
3. Do not write anything on question paper (Except your Er. No.).

Marks

Q1. Imagine there is a tollbooth at a national highway nearby JUET. Cars passing by the booth are expected to pay a 70 rupees toll. Mostly they do, but sometimes a car goes by without paying. The tollbooth keeps track of the number of cars passing, and total amount of money collected. Model this tollbooth with a class called ***Tollbooth*** that includes two static data members namely ***total_number_of_cars*** and ***total_amount*** to store data about total number of cars passed and total amount of money collected, respectively. Define a static member function called ***payingCar()*** that increments the car total and adds 70 rupees to the cash total. Another static member function, called ***notpayingCar()*** that increments the car total but adds nothing to the cash total. Also, define a static member function called ***display()*** to print total number of cars and total amount. Make appropriate member functions const. Write the main function without making any object of the class. Use do-while loop repeatedly to show following menu to the user: **[05]**

Enter 1 for paying car (call to payingCar() function)
Enter 2 for non paying car (call to notpayingCar() function)
Enter 3 for Display (call to display() function)
Enter 4 to exit

Q2. Define a class ***Time*** with three data members namely hours, minutes and seconds. Declare a parameterized constructor to initialize these data members. Write a constant member function called ***showdata()*** to print the values of data members. Write a member function called ***HoursMinutesSeconds_to_Seconds()*** that converts and returns the time given in hours, minutes, and seconds to equivalent time in seconds. In the main function take the values of time in hours, minutes, and seconds from the user and then create an object by sending these values as arguments. Call ***showdata()*** and ***HoursMinutesSeconds_to_Seconds()*** functions suitably from the main function. **[05]**

Q3. The following code tries to concatenate two strings by overloading binary addition(+) operator. Complete the given code by writing a friend operator function for binary addition(+) operator and test the code by writing suitable code in the main function. The operator function should take two objects of given class and return the object of same class. The main function should declare two objects s1 and s2 with values "Object oriented programming" and "Operator overloading" respectively. The operator function should store "Object oriented programming: Operator overloading" in the object after concatenation. The code should stop execution if the length of string passed in s1 and s2 exceeds the macro max (defined in following code) after printing the message "String Overflow" from the operator function. Call the display function from main to print the string values present in the objects. [Hint: use standard string functions(strlen(), strcat(), strcpy(), etc.) to save time.] **[05]**

```
#include <iostream>
#include<cstring>
using namespace std;
#define max 100
class String
{
                char str[max];
        public:
                String()
                {strcpy(str , "");}
                String( char s[])
                {strcpy(str , s);}
                void display()
                {cout<<str<<endl;}
                //Declare friend operator function here
};
//Define the friend operator function here
int main()
{
        //Complete the main by yourself.
}
```

Q4. Define the following terms with the help of example(s): **[05]**

(a) Constructor and Destructor

(b) new and delete operators

Q5. Predict the output/error of the following questions: **[05]**

(a)
```
#include<iostream>
using namespace std;
class Point
{
  public:
        Point()
        { cout << "Constructor called"; }
};
int main()
{
  Point t1, *t2;
  return 0;
}
```

(b)
```
#include <iostream>
using namespace std;
class Test
{
        static int x;
        public:
        Test() { x++; }
        static int getX()
        {return x;}
};
int Test::x = 0;
int main()
{
   cout << Test::getX() << " ";
   Test t[5];
   cout << Test::getX();
}
```

(c)
```
#include <iostream>
using namespace std;
class A
{
        public:

        A(){cout<<"Constructor of A
called"<<endl;}

        ~A(){cout<<"Destructor of A
called"<<endl;}
};
class B
{
        public:
        B(){cout<<"Constructor of B
called"<<endl;}

        ~B(){cout<<"Destructor of B
called"<<endl;}
};
int main()
{
        A a;
        B b;
        return 0;
}
```

(d)
```
#include<iostream>
using namespace std;
class Point
{
  public:
     Point()
     { cout << "c1 "; }
     Point(const Point& p)
     { cout<<"c2 ";}
     Point& operator =(const Point& c)
     { cout<<"c3 ";
       return *this;}
};
int main()
{
  Point *t1, *t2;
  t1 = new Point();  c1
  t2 = new Point(*t1);  c1
  Point t3 = *t1;   c1
  Point t4;
  t4 = t3;
  return 0;
}
```

(e)

```
#include<iostream>
using namespace std;
class Test
{
private:
 int x;
 int y;
public:
 Test(int x = 0, int y = 0)
 {
        this->x = x;
        this->y = y;
 }
 static void fun1()
 {
        cout << "Inside fun1()";
 }
 static void fun2()
 {
        cout << "Inside fun2()";
        this->fun1();
 }
};

int main()
{
 Test obj;
 obj.fun2();
 return 0;
}
```