VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Slot: A1

School of Computer Science Engineering and Information Systems

Winter Semester 2023-2024

Continuous Assessment Test – II

Programme Name & Branch: B.Tech IT

Course Name & code : Structured and Object Oriented Programming,x BCSE 102L

Class Number (s) : VL2023240503283, VL2023240503282, VL2023240503270

Faculty Name (s) : Dr. Suganya P, Dr. Chandra Moulishwaran S, Dr. Rampriya R S

Exam Duration : 90 Min. **Maximum Marks: 50**

Answer Key

Q.No.	Question	Max Marks

1.	Write C program to accept batting information of cricket team using structure and dynamic memory allocation. It contains player name and runs scored by player. Calculate total runs scored by cricket team. Display the runs of the individual players and the total runs scored by the team.
	
	Main Logic:
	
	#include<stdio.h>
	struct player
	{
	char name[20];
	int runs;
	};
	int main()
	{
	int i,s=0;
	struct player a[11]; //a[11] - no. of players
	printf("Enter Name of Player Runs Scored \n");
	printf("----------\n\t");
	for(i=0;i<=10;i++)
	{
	scanf("%s",a[i].name);
	scanf("%d",&a[i].runs);
	printf("\t");
	}
	for(i=0;i<=10;i++)
	s=s+a[i].runs;
	printf("\n----------\n");
	printf("Total Runs Scored by Team: %d",s);
	return 0;
	}
		10

	Mark Distribution: To **Accept batting information** of cricket team (player name and runs scored by player) **using structure** and dynamic memory allocation. **4 M** **Calculate total runs** scored by cricket team. **3 M** **Display the runs** of the individual players and the **total runs** scored by the team. **3 M**	

2.	A fingerprint machine is kept in a school. All the students are required to record their fingerprint on their arrival. Expected arrival time is common to all the students. Given the details of students, their arrival time on a particular day and expected arrival time, design an OOP model and write a C++ code to print the roll no and the name of the students who came late and calculate fine amount. For every one minute of late arrival, 50 rupees is charged as fine.
	
	Main Logic:
	
	#include<iostream>
	using namespace std;
	class biometric
	{
	public:
	int hour1,hour2,minute1,minute2;
	int diff_minute,diff_hour;
	int fine;
	int fineamt;
	double late;
	biometric()
	{
	cout<<"\nEnter the arrival time:";
	cout << "Enter time period 1" << endl;
	cout << "Enter hours, minutes:"<< endl;
	cin >> hour1 >> minute1;
	cout<<"\nEnter the expected time:";
	cout << "Enter hours, minutes: "<< endl;
	cin >> hour2 >> minute2;
	fine = 50;
	}
	public:
	void print()
	{
	if(minute2 > minute1) {
	hour1--;
	minute1 += 60;
	}
	diff_minute = minute1 - minute2;
	diff_hour = hour1 - hour2;
	late= (diff_hour * 60)+diff_minute;
		10

	fineamt=fine*late;
	cout<<fineamt;
	
	}
	
	};
	main()
	{
	biometric b;
	b.print();
	}
	
	Mark Distribution:
	Declare Class, Data Members, Member function 1 to get Student details, arrival time and expected time - **4 M**
	Member function 2 to calculate fine amount – **4 M**
	Main contains object declaration and accessing members – **2 M**	

3.	Write a C++ program to find Strong Numbers within a range of numbers by using friend function in C++.
	[A strong number is one in which the factorial of the digits equals the number itself. Ex: 145 => 1! + 4! + 5! = 1 + 24 + 120 = 145]
	
	Main Logic:
	
	#include<iostream>
	using namespace std;
	class Strong_Numbers
	{
	protected :
	int i, n, n1;
	int s1 , j, k;
	int en, sn;
	long factorial;
	public :
	input()
	{
	s1 = 0;
	factorial = 0;
	cout<<"Please input the starting range of number : "<<endl;
	cin>>sn;
	cout<<"Please input the ending range of number : "<<endl;
	cin>>en;
	}
	friend int disp(Strong_Numbers);
	} ;
	int disp(Strong_Numbers a)
	{
	cout<<"The Strong numbers are: "<<endl;
		10

```
for (a.k = a.sn; a.k <= a.en; a.k++)
{
   a.n1=a.k;
   a.s1 = 0;
   for (a.j = a.k; a.j > 0; a.j = a.j / 10)
   {
      a.factorial = 1;
      for (a.i = 1; a.i <= a.j % 10; a.i++)
      {
         a. factorial = a. factorial * a.i;
      }
      a.s1 = a.s1 + a. factorial;
   }
   if (a.s1 == a.n1)
   {
      cout << a.n1 << "  ";
}
}
cout<<endl;
}
int main()
{
Strong_Numbers a;
a.input();
disp(a);
}
```

Mark Distribution:

Declare Class, Data Members, Member function to get start and end range of numbers and Friend Function – **4 M**

Define Friend Function to find strong no's between the entered range – **4 M**

Main contains object declaration and accessing members – **2 M**

4. Create a base class Patient (pat-name, age, sex) and IPD (ward-no, bed-no, charge-per-day). Derive a class IPD-patient from these two base classes with no-of-days-admitted attribute. Print the patient details, IPD details, no of days admitted and the charge for the no of days admitted. — 10

Main Logic:

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

class Patient    //Base Class
{
      public:
      char patient_name[100];
      int age;
      char sex[50];
```

```
    public:
        void accept_patient_details()
        {
            cout<<"\n Enter Patient Details";
            cout<<"\n Patient Name: ";
            cin>>patient_name;
            cout<<"\n Patient Age:  ";
            cin>>age;
            cout<<"\n Sex:  ";
            cin>>sex;
        }
        void display_patient_details()
        {

            cout<<"\n Displaying Patient Details";
            cout<<"\n Patient Name  :  "<<patient_name;
            cout<<"\n Patient Age   :  "<<age;
            cout<<"\n Sex           :  "<<sex;
        }
};
class IPD    //Base Class
{
   public:
    int ward_no;
    int bed_no;
    int charge_per_day;
  public:
    void accept_ipd_details()
    {
        cout<<"\n Enter IPD Details ";
        cout<<"\n Ward No.        :  ";
        cin>>ward_no;
        cout<<"\n Bed No.         :  ";
        cin>>bed_no;
        cout<<"\n Charge Per Day  :  ";
        cin>>charge_per_day;
    }
    void display_ipd_details()
    {
        cout<<"\n Displaying IPD Details";
        cout<<"\n Ward No.         :  "<<ward_no;
        cout<<"\n Bed No.          :  "<<bed_no;
        cout<<"\n Charge Per Day   :  "<<charge_per_day;
    }
};
//Class IPDPatient is derived from Class IPD and Class Patient
class IPDPatient : public IPD, public Patient
{
    int no_of_days_admitted;
```

```
   public:
      void accept_ipd_patient_details()
      {
            accept_patient_details();
            accept_ipd_details();
            cout<<"\nEnter No. of Days Admitted   :  ";
            cin>>no_of_days_admitted;
      }
      void display_ipd_patient_details()
      {
            display_patient_details();
            display_ipd_details();
            cout<<"\n No. of Days Admitted:  "<<no_of_days_admitted;
            cout<<"\nBill:"<<charge_per_day*no_of_days_admitted;
      }
};
int main()
{
      IPDPatient *ipdt;    // Object ipdt is created of class IPDPatient <-- Child
Class
      int i,cnt;
      cout<<"\n Enter No. of Patient Details You Want  :  ";
      cin>>cnt;
      ipdt=new IPDPatient[cnt];
      for(i=0;i<cnt;i++)
      {
            ipdt[i].accept_ipd_patient_details();

      }
      for(i=0;i<cnt;i++)
         ipdt[i].display_ipd_patient_details();
      return 0;
}
```

Mark Distribution:

Implement Multiple Inheritance – **2 M**
Get & Display Patient Details using Patient Class – **2 M**
Get & Display IPD Details using IPD Class with Charge/Day – **2 M**
Get & Display IPD-Patient Details using IPD-Patient Class with No. od Days admitted – **2 M**
In Main() Declare Array of Object for getting no, of patient details and access the member of derived class and generate bill - **2 M**

5.	a)Find the output of the following:	

a)

```
#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();
}
```

b)

```
#include<iostream>
using namespace std;
class Test
{
private:
   static int count;
public:
   Test& fun();
};
int Test::count = 0;
Test& Test::fun()
{
   Test::count++;
   cout << Test::count << " ";
   return *this;
}
int main()
{
   Test t;
   t.fun().fun().fun().fun();
   return 0;
}
```

c)

```
#include<iostream>
using namespace std;
class Point {public:
Point() {
cout << "Normal Constructor called\n";
}
Point(const Point &t) {
 cout << "Copy constructor called\n";
}
};
int main(){
   Point *t1, *t2;
   t1 = new Point();
   t2 = new Point(*t1);
   Point t3 = *t1;
   Point t4;
   t4 = t3;
   return 0;
}
```

6

Answer

a) 0 5 - **2 M**

b) 1 2 3 4 - **2 M**

c) Normal Constructor called
Copy constructor called
Copy constructor called
Normal Constructor called - **2 M**

4

b)Answer the questions (i) to (iv) based on the following:

```
class PUBLISHER {
char Pub[12];
double Turnover;
protected:
void Register();
public:
PUBLISHER();
void Enter();
void Display();
};
```

```
class BRANCH {
char CITY[20];
protected:
float Employees;
public:
BRANCH();
void Haveit();
void Giveit();
};
```

```
class AUTHOR:private BRANCH,public PUBLISHER
{
int Acode;
char Aname[20];
float Amount;
public:
AUTHOR();
void Start();
void Show();
};
```

i) Write the names of data members, which are accessible from objects belonging to class AUTHOR.

ii) Write the names of all the member functions which are accessible from objects belonging to class BRANCH.

iii) Write the names of all the members which are accessible from member functions of class AUTHOR.

iv) How many bytes will be required by an object belonging to class AUTHOR?

Answer

i) amount, aname, acode, employees – **1 M**

	ii) haveit(), giveit() – **1 M** iii) amount, aname, acode, employees – **1 M** iv) 72 – **1 M**	