Name : Mukul Kumar

Enrollment number : 201B162

Batch : B5

Subject : OOP

Assignment number : 2

Semester : 2

Mukul Kumar (201B162) B5 (BX2)

Assignment 2

Q1

Sol^n
```
#include <iostream>
using std::cout;
using std::endl;
class vector
{
    float *ptr;
    ~~[illegible]~~ int size=0;
  public:
    vector() {}
    vector(const vector &ref)
    {
        this->size = ref.size
        ptr = (float*) malloc (size *sizeof(float));
        for (int i=0 ; i<size ; i++)
        {
            *(this->ptr+i) = *(ref.ptr+i);
        }
    }
    void pushback (int x)
    {
        size++;
        if (size==1)
            ptr = (float*) malloc (size*sizeof(float));
        else
            ptr = (float*) realloc (ptr, size*sizeof(float));
        *(ptr+size -1) = x;
    }
    void modifybyindex (int i, ~~[illegible]~~ float x)
    {
        *(ptr+i) = x;
    }
```

```
void   modifybyvalue(~~[illegible]~~ float i, ~~[illegible]~~ float x)
{
    int index;
    for (int j=0; j<size ; j++)
    {
        if (i== *(ptr +j))
        {
            index = j;   break;
        }
    }
    modifyby~~B~~index(index , x);
}
vector  operator*(int a)
{
    vector  temp (*this);
    for (int i=0 ; i<size ; i++)
    {
        *(temp.ptr+i) = *(temp.ptr+i )*a;
    }
    return temp;
}
void  display()
{
    cout<<"{ ";
    for (int i=0; i<size; i++)
    {
        cout << *(ptr+i);
        if (i!=size -1)
        {
            cout << "  ,";
        }
    }
```

```
        cout << "  }" << endl;
    }
};
~~[illegible]~~     ~~[illegible]~~
int  main ()
{
  vector  a, b;
    a. pushback (1.4);
    a. push back (2.6);
    a. pushback (3.8);
    a. pushback (4);
    a. display ();
    a. modifyby index (3, 5.4);
    a. display ();
    a. modify by value (5.4, 6.6);
    a. display ();
     b = a * 2.5;
     a. display ();
     b. display ();

     return 0;
 }
```

Q 2

Solⁿ To determine size of derived class we can use sizeof (derived_class_name) which will return an integer value representing its size.

There are many factors that determine the size of derived class in C++. They are:-

1. Size of all non-static data member :-
→ Only non-static data members will be counted for calculating size of class/object.

```
class A {
  private:
     float imem1;
     const int imem2;
     static int imem3;
      char imem4;
 };
```

For object of class A, the size will be size of float imem1 + size of int imem2 + size of char imem4;

2. Order of Data members
→ class c {

```
        char c;
        int int1;
        int int2;
        int i;
        long l;
        short s;
 };
```

The size of class is 24 bytes. Even though char c will ~~b~~ consume only 1 byte, 4 bytes will be allocated for it and remaining 3 bytes are ~~used~~ wasted
If we write class in different order.

```
class c {
        int int1;
        int int2;
        int i;
```

```
    long l;
    short s;
    char c;
};
```

Now size of this class is 20 bytes

3. Byte Alignment or Byte padding
As mentioned above, if we specify 1 byte alignment, the size of the class above (class C) will be 19 in both cases.

4. Size of its immediate base class
The size of a class also includes size of ~~#~~ its immediate base class.
For ex

```
        class B {
            int imem1;
            int imem2;
        };
        class D : public B {
                int imem;
        }
```

In this ~~class~~ case, size of (D) is also include the size of A, so it will be 12 bytes.

5. Mode of Inheritance :-
In c++, sometimes we have to use virtual inheritance for some reasons, when we use virtual inheritance, there will be the overhead of 4 bytes for a virtual ~~class~~ base class pointer in that class.

Q.)

Soln
```
#include <iostream>
using std :: cout;
using std :: endl;
class FLOAT
{
    float x;
   public:
    FLOAT() {}
    FLOAT(float x) : x(x) {}
    FLOAT operator +(FLOAT y)
    {
       FLOAT temp;
       temp.x = x + y.x;
       return temp;
    }
    FLOAT operator -(FLOAT y)
     {
        FLOAT temp;
        temp.x = x - y.x;
        return temp;
     }
    FLOAT operator *(FLOAT y)
     {
        FLOAT temp;
        temp.x = x * y.x;
        return temp;
     }
     FLOAT operator /(FLOAT y)
      {
        FLOAT temp;
try
{
    if (y.x == 0)
    {
        throw y.x;
    }
}
catch (float x)
{
    cout << "division by zero" << endl;
    return *this;
}
temp.x = x / y.x;
return temp;
}
void display()
{
    cout << x << endl;
}
};
int main()
{
    FLOAT a(4), b(0), c,d ,e, f;
    c= a+b;
    c.display();
    d=a-b;
    d. display();
    e = a*b;
    e. display ();
    f = a/b;
    f.display();
    return 0;
}
```

Q.4
Solⁿ.

```
# include <iostream>
using std:: cout;
using std:: endl;
using std:: string;
class student
{
 protected:
    string name;
    int branch_number;
    student() {}
    student(int b, string name): name(name),
                      branch_number(b) {}
 };
class CSE : private student
{
    int CSE_sub1_marks, CSE_sub2_marks,
        CSE_sub3_marks;
  public:
    CSE(){}
    CSE(string name, int m1, int m2, int m3):
    student(1, name), CSE_sub1_marks (m1),
    CSE_sub2_marks (m2), CSE_sub3_marks (m3) {}
    void display()
    {
     cout << "Branch_Number" << branch_number <<
     " name - " << name << " CSE_sub1_marks " <<
     CSE_sub1_marks <<" CSE_sub2_marks " <<
     CSE_sub2_marks <<" CSE_sub3_marks " <<
     CSE_sub3_marks << endl;
     }
  };
```

```
class    ECE : private  student
{
    int   ECE_sub1_marks , ECE_sub2_marks ,
      ~~ECE~~ ECE_sub3_marks;

   public :
        ECE () {}
        ECE (string  name , int  m1 , int  m2 , int  m3) :
        student (1, name ), ECE_sub1_marks (m1),
        ECE_sub2_marks (m2), ECE_sub3_marks (m3) {}
        void  display ()
        {
          cout << "Branch  Number " << branch_number
          << "  name - " << name << " ECE_sub1_marks "<<
          ECE_sub1_marks << " ECE_sub2_marks " <<
          ECE_sub2_marks << " ECE_sub3_marks " <<
          ECE_sub3_marks << endl;
          }
        };
    int  main ()
    {
      CSE  a("hello", 10, 20, 30 ), b("world", 11, 21, 31 ),
           c (" test ", 12, 22, 32);
      ECE  d (" 1st object ", 13, 23, 33) , e ("2nd object",
           14, 24, 34) ;
        a.display ();
        b.display ();
        c.display ();
        d.display ();
        e.display ();
        return 0;
}
```

Question 5 : Let we have to prepare the final result of each student for a particular subject. The final marks are sum of marks obtained by the student in T1, T2, T3, P1, P2, and attendance in theory class. Let there are following classes:

(a) Student: Its data members are student name and roll number and member function is to print the values of the data members.

(b) T1T2T3: Its data members are marks obtained by a student in T1, T2, and T3 and member function is to print the values of the data members.

(c) P1P2: Its data members are marks obtained by a student in P1 and P2 and member function is to print the values of the data members.

(d) Attendance: Data member of this class is the percentage of attendance of a student in the theory class and member function is to print the values of the data member.

(e) Total: Data members of this class are total marks obtained and the grade secured by a student and member function is to print the values of the data members.Apart from that, there are two other classes:

(a) Faculty: which have no data member but have a member function to assign the marks of T1, T2, T3, P1, P2, and percentage of attendance to each student.

(b) Administration: which have no data member but have member functions to enter the name and roll number of each student, and to calculate the total marks and final grade of each student. Grade 'A' for> 80% marks, 'B' for 70 to 80%, 'C' for 60 to 70%, D for 50 to 60%, F for <50%.Base on above information, do the following:

(i) Write a function which prints the name and grades of all students in the ascending order of the grades. In case of same grade, print all respective names in alphabetical order.

(ii) Write a function which prints the name and grades of all students in the alphabetical order of the name of the students.

(iii) Write a function to search the grade of a student based on the first name of the student. In case of multiple entries with same name, print all the names with roll number and respective grades.

Here, consider Student as a base class whose derived classes are T1T2T3, P1P2, and Attendance. Further, consider the class Total which inherits the classes T1T2T3, P1P2, and Attendance as multiple inheritance. Classes Faculty and Administration are independent classes.

Code:

```
#include <iostream>

using std::cout;

using std::endl;

using std::string;
```

```
class student
{
  protected:
            string name;
            int roll_no;

  public:
            void display()
            {
                      cout << "Name : " << this->name << " Roll no : " << roll_no << endl;
            }
};
class T1 : virtual public student
{
  protected:
            int sub_marks[4];

  public:
            void display_marks()
            {
                      cout << " sub1 " << sub_marks[0] << " sub2 " << sub_marks[1] << " sub3 " <<
sub_marks[2] << " sub4 " << sub_marks[3] << endl;
            }
};
class T2 : virtual public student
{
  protected:
            int sub_marks[4];
  public:
            void display_marks()
            {
                        cout << " sub1 " << sub_marks[0] << " sub2 " << sub_marks[1] << " sub3 " <<
sub_marks[2] << " sub4 " << sub_marks[3] << endl;
            }
};
class T3 : virtual public student
{
  protected:
            int sub_marks[4];

  public:
            void display_marks()
            {
                        cout << " sub1 " << sub_marks[0] << " sub2 " << sub_marks[1] << " sub3 " <<
sub_marks[2] << " sub4 " << sub_marks[3] << endl;
            }
};
class P1 : virtual public student
{
  protected:
            int sub_marks[4];

  public:
            void display_marks()
            {
                        cout << " sub1 " << sub_marks[0] << " sub2 " << sub_marks[1] << " sub3 " <<
sub_marks[2] << " sub4 " << sub_marks[3] << endl;
            }
};
```

```
class P2 : virtual public student
{
  protected:
            int sub_marks[4];

  public:
            void display_marks()
            {
                        cout << " sub1 " << sub_marks[0] << " sub2 " << sub_marks[1] << " sub3 " <<
sub_marks[2] << " sub4 " << sub_marks[3] << endl;
            }
};
class Attendance : virtual public student
{
  protected:
            int attendance_percent;

  public:
            void display()
            {
                        cout << "Attendance pecentage is " << attendance_percent << endl;
            }
};
class Total : public T1, public T2, public T3, public P1, public P2, public Attendance
{
            int total_marks[4] = {0};
            int total_percent;

  public:
            void total_mark()
```

```
    {
        int Attendance_mark;
        if (attendance_percent >= 90)
        {
            Attendance_mark = 5;
        }
        else if (attendance_percent >= 88)
        {
            Attendance_mark = 4;
        }
        else if (attendance_percent >= 86)
        {
            Attendance_mark = 3;
        }
        else if (attendance_percent >= 83)
        {
            Attendance_mark = 2;
        }
        else if (attendance_percent >= 80)
        {
            Attendance_mark = 1;
        }
        else
        {
            Attendance_mark = 0;
        }
        for (int k = 0; k < 4; k++)
        {
            total_marks[k] = T1::sub_marks[k] + T2::sub_marks[k] +
T3::sub_marks[k] + P1::sub_marks[k] + P2::sub_marks[k] + Attendance_mark;
```

```
                }
                total_percent = (total_marks[0] + total_marks[1] + total_marks[2] +
total_marks[3]) / 4;
        }
        friend void studgrade(Total *, int);
        friend void studgrade_lex(Total *, int);
        friend void search(Total *, string, int);
        friend class Faculty;
        friend class Administration;
};

void studgrade(Total *ptr, int size)
{
        cout << endl
                 << "------------" << endl;
        string gradetemp[size][2];
        for (int i = 0; i < size; ++i)
        {
                ptr[i].total_mark();
                if (ptr[i].total_percent > 80)
                {
                        gradetemp[i][0] = "A";
                }
                else if (ptr[i].total_percent > 70)
                {
                        gradetemp[i][0] = "B";
                }
                else if (ptr[i].total_percent > 60)
                {
                        gradetemp[i][0] = "C";
```

```
		}
		else if (ptr[i].total_percent > 50)
		{
			gradetemp[i][0] = "D";
		}
		else if (ptr[i].total_percent <= 50)
		{
			gradetemp[i][0] = "F";
		}
		gradetemp[i][1] = ptr[i].name;
		//cout << gradetemp[i][0] << "\t" << gradetemp[i][1] << endl;
	}
	for (int i = 0; i < size; ++i)
	{
		for (int j = 0; j < size - i; ++j)
		{
			if (gradetemp[j][0] < gradetemp[j + 1][0])
			{
				string temp1, temp2;
				temp1 = gradetemp[j][0];
				temp2 = gradetemp[j][1];
				gradetemp[j][0] = gradetemp[j + 1][0];
				gradetemp[j][1] = gradetemp[j + 1][1];
				gradetemp[j + 1][0] = temp1;
				gradetemp[j + 1][1] = temp2;
			}
		}
		//cout << gradetemp[i][0] << "\t" << gradetemp[i][1] << endl;
	}
	for (int i = 0; i < size; ++i)
```

```
        {
                for (int j = 0; j < size - i - 1; ++j)
                {
                        if (gradetemp[j][0] == gradetemp[j + 1][0] && gradetemp[j][1] <
gradetemp[j + 1][1])
                        {
                                string temp1, temp2;
                                temp1 = gradetemp[j][0];
                                temp2 = gradetemp[j][1];
                                gradetemp[j][0] = gradetemp[j + 1][0];
                                gradetemp[j][1] = gradetemp[j + 1][1];
                                gradetemp[j + 1][0] = temp1;
                                gradetemp[j + 1][1] = temp2;
                        }
                }
        }
        for (int i = size-1; i >= 0; i--)
        {
                //cout << "a" <<endl;
                cout << gradetemp[i][0] << "\t" << gradetemp[i][1] << endl;
        }
        cout << endl
                 << "------------" << endl;
}
void studgrade_lex(Total *ptr, int size)
{
        string gradetemp[size][2];
        for (int i = 0; i < size; ++i)
        {
                if (ptr[i].total_percent > 80)
```

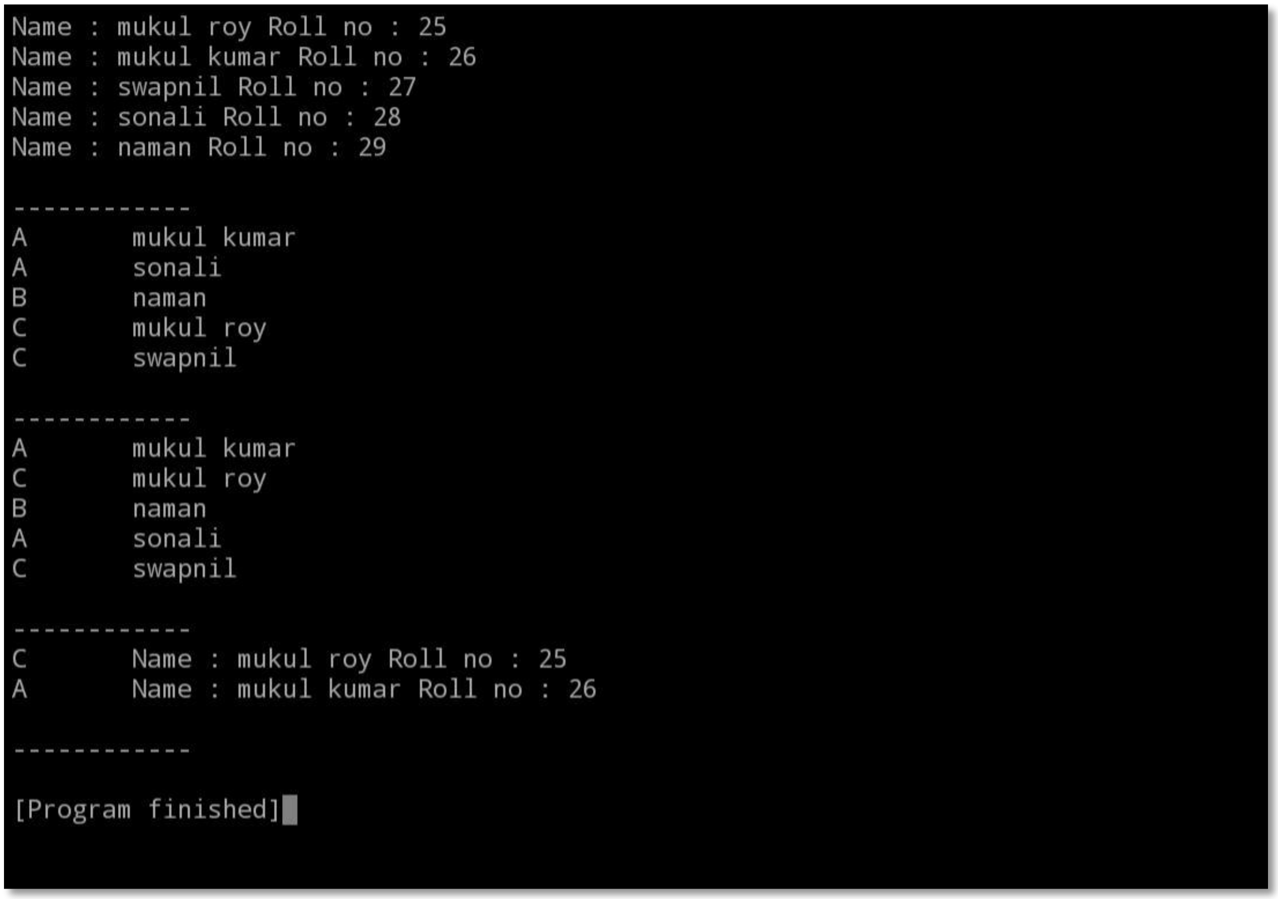
```
		{
			gradetemp[i][0] = "A";
		}
		else if (ptr[i].total_percent > 70)
		{
			gradetemp[i][0] = "B";
		}
		else if (ptr[i].total_percent > 60)
		{
			gradetemp[i][0] = "C";
		}
		else if (ptr[i].total_percent > 50)
		{
			gradetemp[i][0] = "D";
		}
		else if (ptr[i].total_percent <= 50)
		{
			gradetemp[i][0] = "F";
		}
		gradetemp[i][1] = ptr[i].name;
	}

	for (int i = 0; i < size; ++i)
	{
		for (int j = 0; j < size - i - 1; ++j)
		{
			if (gradetemp[j][1] > gradetemp[j + 1][1])
			{
				string temp1, temp2;
				temp1 = gradetemp[j][0];
```

```
                temp2 = gradetemp[j][1];
                gradetemp[j][0] = gradetemp[j + 1][0];
                gradetemp[j][1] = gradetemp[j + 1][1];
                gradetemp[j + 1][0] = temp1;
                gradetemp[j + 1][1] = temp2;
            }
        }
    }
    for (int i = 0; i < size; i++)
    {
        cout << gradetemp[i][0] << "\t" << gradetemp[i][1] << endl;
    }
    cout << endl
         << "------------" << endl;
}
void search(Total *ptr, string temp, int size)
{
    int space_index_temp, space_index;
    for (int i = 0; i < temp.length(); i++)
    {
        if (temp.at(i) == ' ')
        {
            //cout << i <<endl;
            space_index_temp = i;
            break;
        }
    }
    for (int k = 0; k < size; k++)
    {
        string temproary = ptr[k].name;
```

```
for (int i = 0; i < temproary.length(); i++)
{
        if (temproary.at(i) == ' ')
        {
                //cout << i <<endl;
                space_index = i;
                break;
        }
}
if (!(temproary.compare(0, space_index, temp, 0, space_index_temp)))
{
        if (ptr[k].total_percent > 80)
        {
                cout << "A\t";
        }
        else if (ptr[k].total_percent > 70)
        {
                cout << "B\t";
        }
        else if (ptr[k].total_percent > 60)
        {
                cout << "C\t";
        }
        else if (ptr[k].total_percent > 50)
        {
                cout << "D\t";
        }
        else if (ptr[k].total_percent <= 50)
        {
                cout << "F\t";
```

```
                        }
                        ptr[k].student::display();
                }
        }
        cout << endl
                 << "------------" << endl;
}

class Faculty
{
  public:
        void addresult(Total &obj, int T1_1, int T1_2, int T1_3, int T1_4, int T2_1, int T2_2, int
T2_3, int T2_4, int T3_1, int T3_2, int T3_3, int T3_4, int P1_1, int P1_2, int P1_3, int P1_4, int
P2_1, int P2_2, int P2_3, int P2_4, int Attend)
        {
                obj.T1::sub_marks[0] = T1_1;
                obj.T1::sub_marks[1] = T1_2;
                obj.T1::sub_marks[2] = T1_3;
                obj.T1::sub_marks[3] = T1_4;
                obj.T2::sub_marks[0] = T2_1;
                obj.T2::sub_marks[1] = T2_2;
                obj.T2::sub_marks[2] = T2_3;
                obj.T2::sub_marks[3] = T2_4;
                obj.T3::sub_marks[0] = T3_1;
                obj.T3::sub_marks[1] = T3_2;
                obj.T3::sub_marks[2] = T3_3;
                obj.T3::sub_marks[3] = T3_4;
                obj.P1::sub_marks[0] = P1_1;
                obj.P1::sub_marks[1] = P1_2;
                obj.P1::sub_marks[2] = P1_3;
                obj.P1::sub_marks[3] = P1_4;
                obj.P2::sub_marks[0] = P2_1;
                obj.P2::sub_marks[1] = P2_2;
                obj.P2::sub_marks[2] = P2_3;
                obj.P2::sub_marks[3] = P2_4;
                obj.attendance_percent = Attend;
        }
};
class Administration
{
  public:
        void addstudentdetail(Total &obj, int roll, string name)
        {
                obj.name = name;
                obj.roll_no = roll;
        }
};
int main()
{
        Total t[5];
        Faculty teacher1;
        Administration admin1;
        admin1.addstudentdetail(t[0], 25, "mukul roy");
        teacher1.addresult(t[0], 10, 11, 12, 13, 19, 20, 21, 22, 8, 9, 10, 11, 7, 8, 7, 8, 12, 13, 12,
13, 91);
        admin1.addstudentdetail(t[1], 26, "mukul kumar");
        teacher1.addresult(t[1], 10, 11, 12, 13, 19, 20, 21, 22, 28, 29, 30, 31, 7, 8, 7, 8, 12, 13, 12,
13, 91);
        admin1.addstudentdetail(t[2], 27, "swapnil");
        teacher1.addresult(t[2], 10, 11, 12, 13, 19, 20, 21, 22, 8, 9, 10, 11, 7, 8, 7, 8, 12, 13, 12,
13, 91);
        admin1.addstudentdetail(t[3], 28, "sonali");
```

```
	teacher1.addresult(t[3], 10, 11, 12, 13, 19, 20, 21, 22, 28, 29, 30, 31, 7, 8, 7, 8, 12, 13, 12, 
13, 91);

	admin1.addstudentdetail(t[4], 29, "naman");

	teacher1.addresult(t[4], 10, 11, 12, 1, 1, 0, 21, 22, 28, 29, 30, 31, 7, 8, 7, 8, 12, 13, 12, 13, 
91);

	for (int i = 0; i < 5; i++)

	{

		t[i].student::display();

	}


	studgrade(t, 5);

	studgrade_lex(t, 5);

	search(t, "mukul", 5);

	return 0;

}
```



```
Name : mukul roy Roll no : 25
Name : mukul kumar Roll no : 26
Name : swapnil Roll no : 27
Name : sonali Roll no : 28
Name : naman Roll no : 29

------------
A       mukul kumar
A       sonali
B       naman
C       mukul roy
C       swapnil

------------
A       mukul kumar
C       mukul roy
B       naman
A       sonali
C       swapnil

------------
C       Name : mukul roy Roll no : 25
A       Name : mukul kumar Roll no : 26

------------

[Program finished]
```

Question 6 : Consider a case of single inheritance where Landline phone is a base class and Mobile phone is the derived class. Both the classes are as follow:

(a) Landline: It has subscriber name and number as data members. The member functions are to provide the features of calling on a subscriber's number and receiving a call.

Void call (int sub_number);

Void receive();

(b) Mobile: Apart from inheriting the features of a Landline phone, it provides following additional features:

(i) Maintaining a phonebook to save the name and phone number of friends and relatives. For this, a data member of type array of strings has to be added.

(ii) Calling to a subscriber with its name.

Void call (string sub_name);

This function first searches the name of the subscriber to be called in the phonebook to find the corresponding phone number and then, invokes the function “void call (int sub_number)” by passing the searched phone number as argument.

(iii) Maintaining a list of last 20 dialled numbers. For this, a data member of type array of 20 integers has to be added. An entry will be made to this array each time whenever call() function will be invoked. In case of 21th entry to the array, the earliest entry will be replaced with the latest entry.

(iv) Calling on a number from the list of dialled numbers. This function first displays the list of dialled numbers and provides an option to choose a phone number from the list to which a call has to be made. Finally, it invokes call() function and passes the chosen phone number as an argument.

Finally, write the main program to show the features of each class.

Code :

```
#include <iostream>

using std::cin;

using std::cout;

using std::endl;

using std::string;

typedef struct phonebook

{

        string name;

        long long int number;

} phonebook;
```

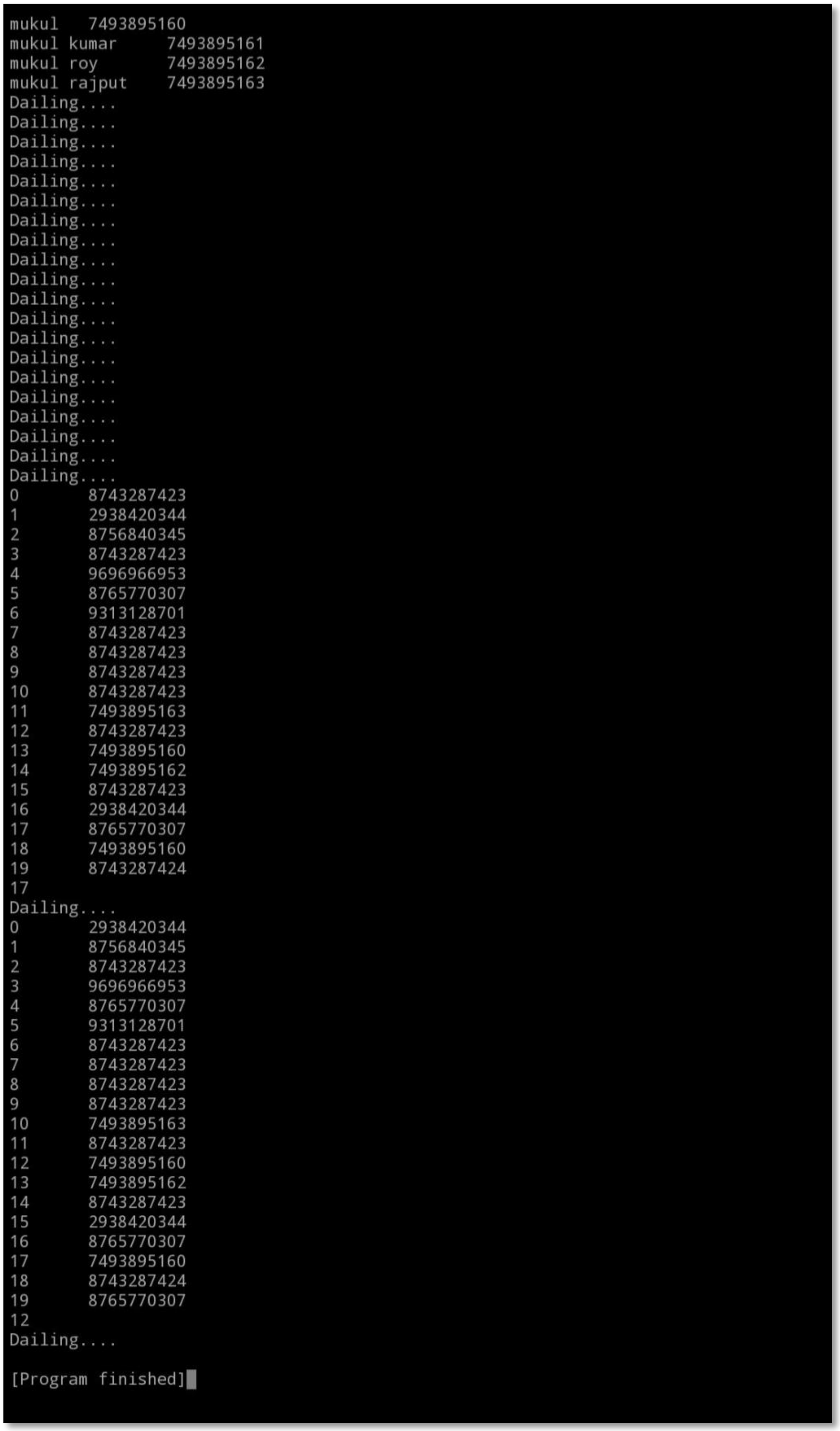
```
typedef struct record
{
        string type;
        long long int number;
} record;

class landline
{
 private:
        record *records_ptr;
        int record_size = 0;
        int flag = 0;

 protected:
        string name;
        long long int number;

 public:
        void sub_data(string name, long long int number)
        {
                this->name = name;
                this->number = number;
        }
        void call(long long int sub_number)
        {
                cout << "Dailing...." << endl;
                if (record_size == 0)
                {
                        record_size++;
                        records_ptr = (record *)malloc(record_size * sizeof(record));
```

```
                                records_ptr[record_size - 1].number = sub_number;
                                records_ptr[record_size - 1].type = "incoming";
                        }
                        else if (record_size <= 20 && flag == 0)
                        {
                                record_size++;
                                records_ptr = (record *)realloc(records_ptr, record_size *
sizeof(record));
                                records_ptr[record_size - 1].number = sub_number;
                                records_ptr[record_size - 1].type = "incoming";
                                if (record_size == 20)
                                {
                                        flag = 1;
                                }
                        }
                        else
                        {
                                record *temp_records_ptr;
                                temp_records_ptr = (record *)malloc(record_size * sizeof(record));
                                for (int i = 0; i < record_size - 1; i++)
                                {
                                        temp_records_ptr[i].number = records_ptr[i + 1].number;
                                        temp_records_ptr[i].type = records_ptr[i + 1].type;
                                }
                                temp_records_ptr[record_size - 1].number = sub_number;
                                temp_records_ptr[record_size - 1].type = "incoming";
                                records_ptr = temp_records_ptr;
                        }
                }
                void recieve(long long int sub_number)
```

```
        {
                if (record_size == 0)
                {
                        record_size++;
                        records_ptr = (record *)malloc(record_size * sizeof(record));
                        records_ptr[record_size - 1].number = sub_number;
                        records_ptr[record_size - 1].type = "recieving";
                }
                else if (record_size <= 20 && flag == 0)
                {
                        record_size++;
                        records_ptr = (record *)realloc(records_ptr, record_size *
sizeof(record));
                        records_ptr[record_size - 1].number = sub_number;
                        records_ptr[record_size - 1].type = "recieving";
                        if (record_size == 20)
                        {
                                flag = 1;
                        }
                }
                else
                {
                        record *temp_records_ptr;
                        temp_records_ptr = (record *)malloc(record_size * sizeof(record));
                        for (int i = 0; i < record_size - 1; i++)
                        {
                                temp_records_ptr[i].number = records_ptr[i + 1].number;
                                temp_records_ptr[i].type = records_ptr[i + 1].type;
                        }
                        temp_records_ptr[record_size - 1].number = sub_number;
}

                        temp_records_ptr[record_size - 1].type = "recieving";
                        records_ptr = temp_records_ptr;
                }
        }
        void callbyhistory()
        {
                int index;
                for (int i = 0; i < record_size; i++)
                {
                        cout << i << "\t" << records_ptr[i].number << endl;
                }
                cin >> index;
                call(records_ptr[index].number);
        }
};
class mobile : public landline
{
  private:
        int phone_book_size = 0;

  protected:
        phonebook *ptr = NULL;

  public:
  void contact ()
  {
        for (int i = 0;i<phone_book_size;i++)
        {
                cout<<ptr[i].name<<"\t"<<ptr[i].number <<endl;
        }
```

```
}
        void addcontact(string pname, long long int pno)
        {
                if (phone_book_size == 0)
                {
                        ++phone_book_size;
                        ptr = (phonebook *)malloc(phone_book_size * sizeof(phonebook));
                }
                else
                {
                        ++phone_book_size;
                        ptr = (phonebook *)realloc(ptr, phone_book_size * sizeof(phonebook));
                }
                if (ptr != NULL)
                {
                        ptr[phone_book_size - 1].name = pname;
                        ptr[phone_book_size - 1].number = pno;
                }
                else
                {
                        addcontact(pname, pno);
                }
        }
        void call(string sub_name)
        {
                for (int i = 0; i < phone_book_size; i++)
                {
                        if (ptr[i].name == sub_name)
                        {
                                landline::call(ptr[i].number);
                        return;
                    }
                }
                cout << "No Contact named " << sub_name << endl;
                return;
            }
};
int main()
{
            mobile m;
            m.sub_data("mukul",7493895160) ;
            m.addcontact("mukul",7493895160) ;
            m.addcontact("mukul kumar",7493895161) ;
            m.addcontact("mukul roy",7493895162) ;
            m.addcontact("mukul rajput",7493895163) ;
            m.contact();
            m.call("mukul");
            m.call("mukul roy");
            m.landline::call(8743287423);
            m.recieve(2938420344);
            m.landline::call(8756840345);
            m.landline::call(8743287423);
            m.landline::call(8743287423);
            m.landline::call(8743287423);
            m.landline::call(8743287423);
            m.landline::call(8743287423);
            m.landline::call(8743287423);
            m.landline::call(8743287423);
            m.landline::call(8743287423);
            m.call("mukul rajput");
```

```
        m.landline::call(8743287423);

        m.call("mukul");

        m.call("mukul roy");

        m.landline::call(8743287423);

        m.recieve(2938420344);

        m.landline::call(8756840345);

        m.call("mukul");

        m.landline::call(8743287423);

        m.callbyhistory();

        m.callbyhistory();

        return 0;

}
```



```
mukul   7493895160
mukul kumar     7493895161
mukul roy       7493895162
mukul rajput    7493895163
Dailing....
Dailing....
Dailing....
Dailing....
Dailing....
Dailing....
Dailing....
Dailing....
Dailing....
Dailing....
Dailing....
Dailing....
Dailing....
Dailing....
Dailing....
Dailing....
Dailing....
Dailing....
Dailing....
Dailing....
0       8743287423
1       2938420344
2       8756840345
3       8743287423
4       9696966953
5       8765770307
6       9313128701
7       8743287423
8       8743287423
9       8743287423
10      8743287423
11      7493895163
12      8743287423
13      7493895160
14      7493895162
15      8743287423
16      2938420344
17      8765770307
18      7493895160
19      8743287424
17
Dailing....
0       2938420344
1       8756840345
2       8743287423
3       9696966953
4       8765770307
5       9313128701
6       8743287423
7       8743287423
8       8743287423
9       8743287423
10      7493895163
11      8743287423
12      7493895160
13      7493895162
14      8743287423
15      2938420344
16      8765770307
17      7493895160
18      8743287424
19      8765770307
12
Dailing....

[Program finished]
```