MohammedRazaKhan

201b156

Q1

Ans:

```
#include <iostream>
#include <conio.h>
#include <stdio.h>
#include <string.h
using namespace std;
class Vector
{
public:
int x, y, z;

void set_vector()
{
cout << "\nEnter the scaler values of vector quentity \n\n x direction      :";
cin >> x;
cout << "\n\n y direction      :";
cin >> y;
cout << "\n\n z direction      :";
cin >> z;
      }
void modify()
{
```

```
int i;
display();
cout << "\nEnter the direction in which you want to modify vector\n1
x-dimension\n2 y-dimension\n3 z-dimension\n";
cin >> i;
if(i==1)
{
cout << "\nEnter new x    :";
cin >> x;
            }
else if(i==2)
{
cout << "\nEnter new y    :";
cin >> y;
            }
else if(i==3)
{
cout << "\nEnter new z    :";
cin >> z;
            }
display();
        }
void multiply()
{
int scalar;
```

```
cout << "\nEnter scalar quantity to multiply by    :";
cin >> scalar;
x = x * scalar;
y = y * scalar;
z = z * scalar;
display();
        }

void display()
{
cout << "\nEntered vector is    "<< x << "i + " << y << "j + " << z <<
"k";
        }
};

int main()
{
Vector vt;
vt.set_vector();
vt.modify();
vt.multiply();
getch();
return 0;
```

```
}
```

Q 2

Ans:

```
Class A
{
public.
virtual void fun1 ();
}
Class B
{
public.
virtual void fun2();
}
Class C
{
public.
virtual void fun3();
}
Class D   private A, protected B, public C
{
  There is not any virtual function of D class as A, B, C
```

}

Q 3

Ans:

```
#include <iostream>
using namespace std;
class Float
{
float i;
public:
Float()  i(5) {}
Float(float x)  i(x) {}
Float operator+(Float a)
{
Float temp;
temp.i = i + a.i;
return temp;
        }
Float operator-(Float a)
{
Float temp;
temp.i = i - a.i;
return temp;
        }
```

```
Float operator/(float a)
{
Float temp;
temp.i=i a;
return temp;
        }
friend Float operator*(float a, Float b)
{
Float temp;
temp.i=a*b.i;
return temp;
        }
void show()
{
cout<<i<<endl;
        }
};
int main()
{
Float a=10.6, b=5.3, c;
cout<<"a=10.6 b=5.3\n";
c=a+b;
cout<<"a+b=";
c.show();
```

```
c = a-b;
cout<<"a-b = ";
c.show();
c = a  5/3;
cout<<"a  5/3 = ";
c.show();
c = 10.6*b;
cout<<"10.6*b = ";
c.show();
return 0;
}
```

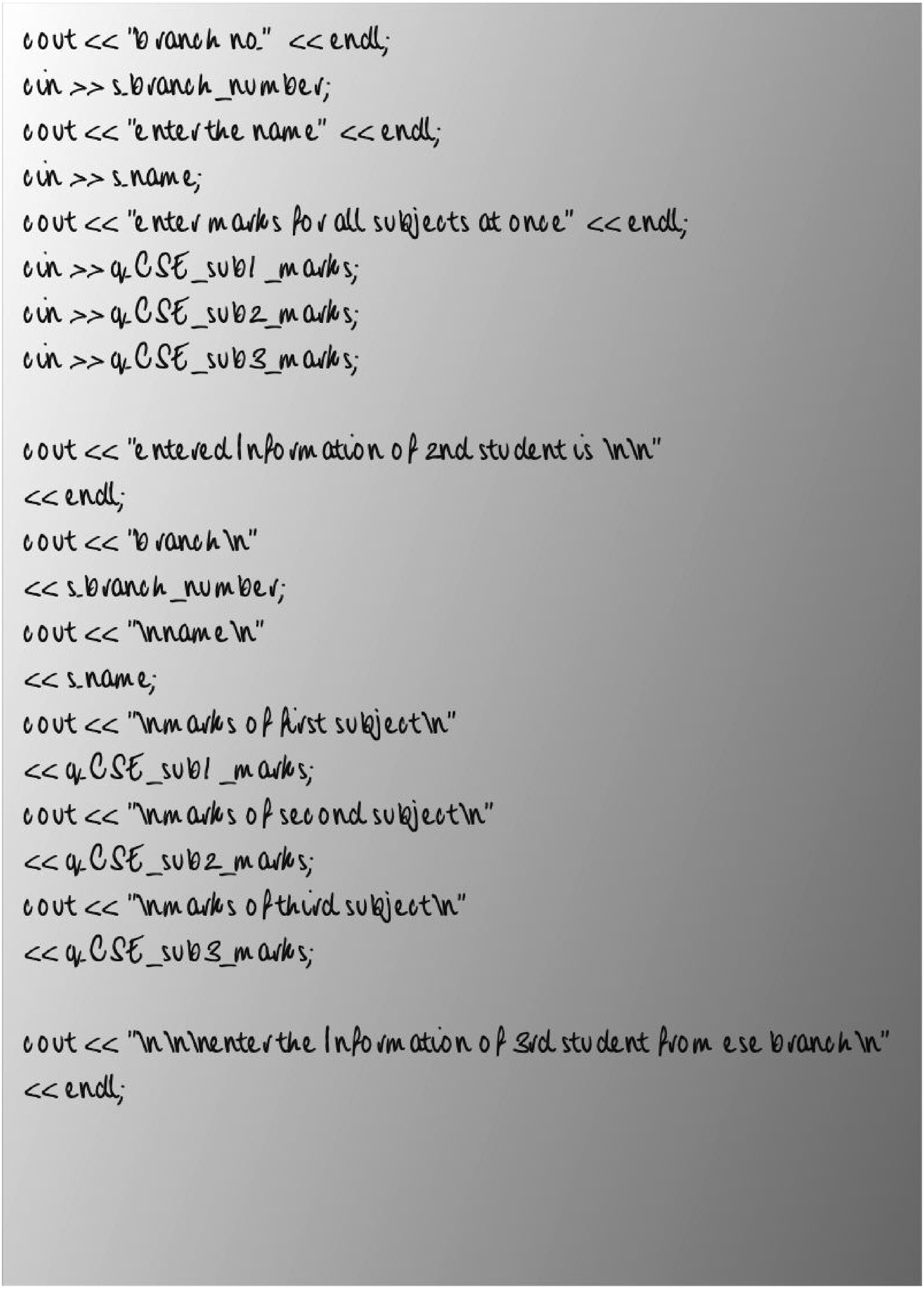
Q4

Ans:

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

class student
{
public:
char name[50];
int branch_number;
};
class cse
{
```

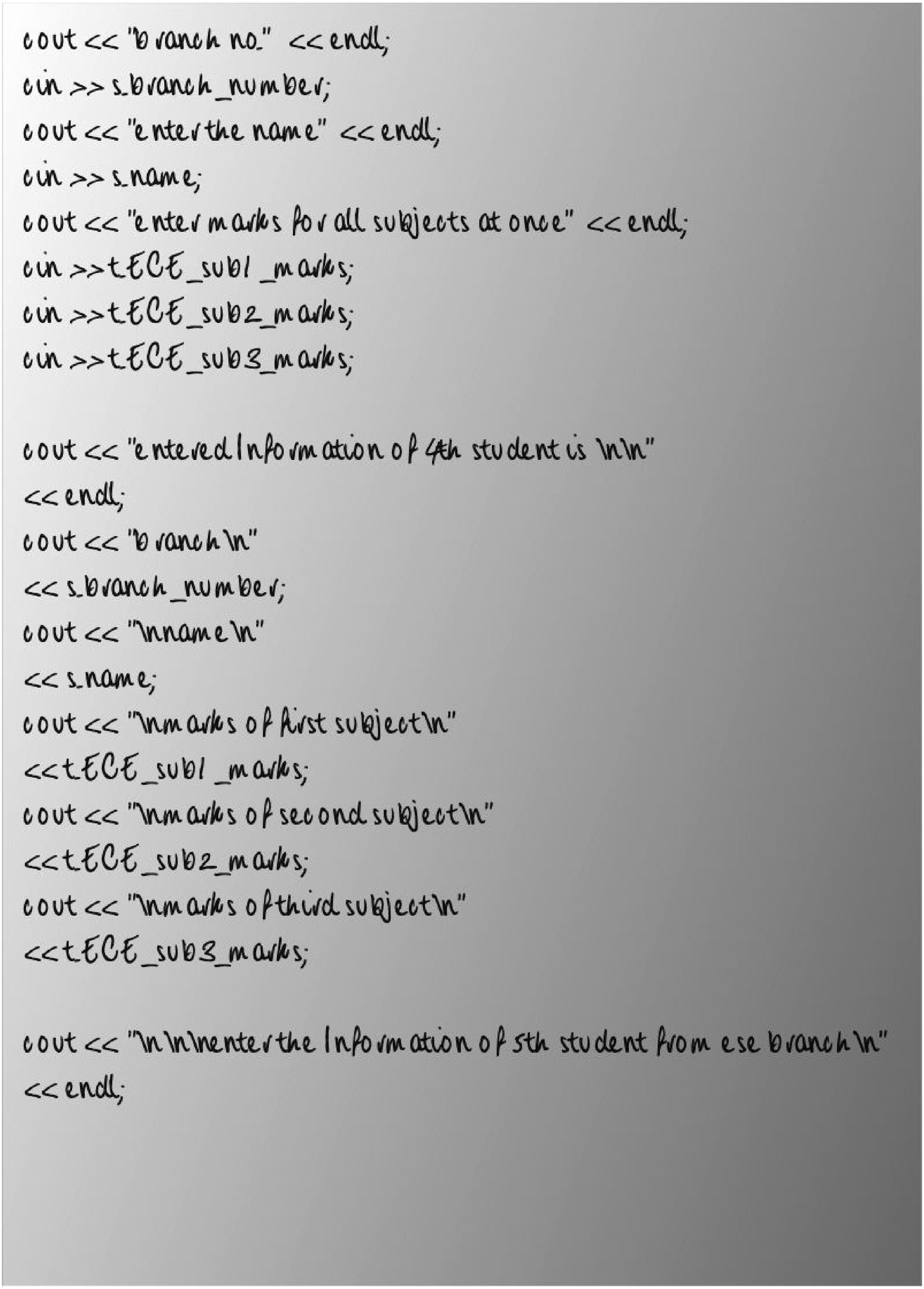
```
public:
int CSE_sub1_marks;
int CSE_sub2_marks;
int CSE_sub3_marks;
};
class ese
{
public:
int ECE_sub1_marks;
int ECE_sub2_marks;
int ECE_sub3_marks;
};

int main()
{
student s;
cse a;
ese t;
cout << "Enter information of students\n"
<<endl;

cout << "enter the Information of 1 st student from cse branch\n"
<<endl;
```

```
cout << "branch no." <<endl;
cin >> s.branch_number;
cout << "enter the name" <<endl;
cin >> s.name;
cout << "enter marks for all subjects at once" <<endl;
cin >> q.CSE_sub1_marks;
cin >> q.CSE_sub2_marks;
cin >> q.CSE_sub3_marks;

cout << "entered Information of 1 st student is \n\n"
<<endl;
cout << "branch\n"
<< s.branch_number;
cout << "\nname\n"
<< s.name;
cout << "\nmarks of first subject\n"
<< q.CSE_sub1_marks;
cout << "\nmarks of second subject\n"
<< q.CSE_sub2_marks;
cout << "\nmarks of third subject\n"
<< q.CSE_sub3_marks;

cout << "\n\n\nenter the Information of 2nd student from cse branch\n"
<<endl;
```



```
cout << "branch no." << endl;
cin >> s.branch_number;
cout << "enter the name" << endl;
cin >> s.name;
cout << "enter marks for all subjects at once" << endl;
cin >> q.CSE_sub1_marks;
cin >> q.CSE_sub2_marks;
cin >> q.CSE_sub3_marks;

cout << "entered Information of 2nd student is \n\n"
<< endl;
cout << "branch \n"
<< s.branch_number;
cout << "\nname \n"
<< s.name;
cout << "\nmarks of first subject\n"
<< q.CSE_sub1_marks;
cout << "\nmarks of second subject\n"
<< q.CSE_sub2_marks;
cout << "\nmarks of third subject\n"
<< q.CSE_sub3_marks;

cout << "\n\n\nenter the Information of 3rd student from ese branch\n"
<< endl;
```

```
cout << "branch no." << endl;
cin >> s.branch_number;
cout << "enter the name" << endl;
cin >> s.name;
cout << "enter marks for all subjects at once" << endl;
cin >>t.ECE_sub1 _marks;
cin >>t.ECE_sub2_marks;
cin >>t.ECE_sub3_marks;

cout << "entered Information of 3rd student is \n\n"
<< endl;
cout << "branch \n"
<< s.branch_number;
cout << "\nname \n"
<< s.name;
cout << "\nmarks of first subject\n"
<<t.ECE_sub1 _marks;
cout << "\nmarks of second subject\n"
<<t.ECE_sub2_marks;
cout << "\nmarks of third subject\n"
<<t.ECE_sub3_marks;

cout << "\n\n\nenter the Information of 4th student from ese branch\n"
<< endl;
```



```
cout << "branch no." <<endl;
cin >>s.branch_number;
cout << "enter the name" <<endl;
cin >>s.name;
cout << "enter marks for all subjects at once" <<endl;
cin >>t.ECE_sub1 _marks;
cin >>t.ECE_sub2_marks;
cin >>t.ECE_sub3_marks;

cout << "entered Information of 4th student is \n\n"
<<endl;
cout << "branch\n"
<<s.branch_number;
cout << "\nname\n"
<<s.name;
cout << "\nmarks of first subject\n"
<<t.ECE_sub1 _marks;
cout << "\nmarks of second subject\n"
<<t.ECE_sub2_marks;
cout << "\nmarks of third subject\n"
<<t.ECE_sub3_marks;

cout << "\n\n\nenter the Information of 5th student from ese branch\n"
<<endl;
```

```
cout << "branch no." << endl;
cin >> s.branch_number;
cout << "enter the name" << endl;
cin >> s.name;
cout << "enter marks for all subjects at once" << endl;
cin >>t.ECE_sub1_marks;
cin >>t.ECE_sub2_marks;
cin >>t.ECE_sub3_marks;

cout << "entered Information of 5th student is \n\n"
<< endl;
cout << "branch \n"
<< s.branch_number;
cout << "\nname \n"
<< s.name;
cout << "\nmarks of first subject\n"
<<t.ECE_sub1_marks;
cout << "\nmarks of second subject\n"
<<t.ECE_sub2_marks;
cout << "\nmarks of third subject\n\n\n"
<<t.ECE_sub3_marks;
return 0;
}
```

Q5

Code:

```
#include<iostream>
#include<string.h>
using namespace std ;
class administration ;
class facality ;
class student
{   friend class administration ;
     friend class facality ;
    string name ;
    int rollno ;

    public  :
     void diplay_student()
     { printf("%-6d",rollno );
       cout<<"                    "<<name ;

     }

};



class T1T2T3 : virtual public student
{ friend class facality ;
 friend class administration ;
  float T1 ,T2 ,T3 ;
   public :
  void display_t1t2t2()
  {
    cout<<"\n marks in :\n";
    cout<<"    T1 = "<<T1<<endl  ;
    cout<<"    T2 = "<<T2<<endl  ;
    cout<<"    T3 = "<<T3<<endl  ;
  }
  };

  class P1P2 :virtual public student
  {  friend class facality ;
   friend class administration ;
    float P1 ,P2 ;
    public :
    void displayp1p2()
    {
    cout<<"\n marks in :\n";
    cout<<"    P1 = "<<P1<<endl  ;
    cout<<"    P2 = "<<P2<<endl  ;
    }

  };

  class attendance :virtual public student
  {
       float attnd_prcnt ;
       public :
       void display_attendence()
       {
         cout<<"\npercentage of attendence : "<<attnd_prcnt ;
       }
  };

  class facality ;
  class total : virtual public T1T2T3 ,virtual public P1P2 , virtual public attendance
  {  friend class facality ;
     friend class administration ;
    float total_marks ;
    char grade ;
     public :
    void display_total()
    {
      cout<<"total marks = "<<total_marks ;
    }
    void display_grade()
    {
      cout<<"      "<<grade<<"                   " ;
    }
  };


class facality
{
  public :
 void get_exam_marks(total & t )
  {
    cout<<"\nenter the marks in :\n" ;
    cout<<"     T1 = ";
    cin>>t.T1 ;
    fflush(stdin) ;
     cout<<"     T2 = ";
    cin>>t.T2 ;
    fflush(stdin) ;
     cout<<"     T3 = ";
    cin>>t.T3 ;
    fflush(stdin) ;
     cout<<"     p1 = ";
    cin>>t.P1  ;
    fflush(stdin) ;
     cout<<"     P2 = ";
    cin>>t.P2 ;
    fflush(stdin) ;



  }
   void sorting1(total * t , int num )
    {  total temp ;

      for(int i= 0 ; i <num ;i++)
      {  for(int j = 1+i ;j < num ; j++ )
           {if (t[j].total_marks > t[i].total_marks)
              { temp = t[i] ;
               t[i] = t[j] ;
               t[j] = temp ;
              }
           }

      }

     /* t[0].diplay_student();
      t[0].display_grade();
      t[1].diplay_student();
      t[1].display_grade();
      t[2].display_grade() ;
      t[2].diplay_student() ;*/
      for(int i =0 ;i< num ;i++)
         { for(int j= i+1 ; j<num ;j++)
           {
               if(t[i].grade == t[j].grade)
               {
                   if(t[i].name > t[j].name)
                     { total temp ;
                      temp = t[i] ;
                      t[i] = t[j] ;
                       t[j] = temp ;
                     }
               }
           }
         }
       cout<<"    GRADE   |         ROLL NO        |                  NAME          \n" ;
        cout<<"______________________________\n" ;
     for (int i =0 ;i < num ; i++)
    {   t[i].display_grade() ;
      t[i].diplay_student() ;

        cout<<"\n____|_________|________________\n" ;


    }
    }

    void sorting2(total * t , int num )
    {
       for(int i =0 ;i< num ;i++)
         { for(int j= i+1 ; j<num ;j++)
           {
                if(t[i].name > t[j].name)
                     { total temp ;
                      temp = t[i] ;
                      t[i] = t[j] ;
                       t[j] = temp ;
                     }

           }
         }
       cout<<"    GRADE   |         ROLL NO        |                  NAME          \n" ;
        cout<<"______________________________\n" ;
     for (int i =0 ;i < num ; i++)
    {   t[i].display_grade() ;
      t[i].diplay_student() ;

        cout<<"\n____|_________|________________\n" ;


    }



    }

    void searching (total * t , int num , string search)
```

```
   {  int k = 0 ;
      int length = search.length() ;
      for(int i = 0 ; i < num ; i ++ )
      {
        if(t[i].name[length] == ' ')
        {
           //t[i].diplay_student() ;
          int j = 0 ;
          for( j = 0 ; j < length ; j ++ )
          { int l= 0 ;
            if( search[j] != t[i].name[j])
              break ;

              if(j == length -1)
              {
                  k ++ ;
                  if(l =0 )
                  {cout<<"    GRADE   |         ROLL NO        |                  NAME         \n" ;
                       cout<<"____________________________\n" ; }
                            t[i].display_grade() ;
                             t[i].diplay_student() ;

                          cout<<"\n_____|________|_______________\n"   ;
               l = 1 ;

              }



          }




        }
      }

      if(k==0)
      {
        cout<<"\nno result found . " ;
      }
   }



};


class administration
{  public :
  void get_student(total& t)
  {  cout<<"enter name : " ;
      fflush(stdin) ;
    getline(cin , t.name ) ;
    fflush(stdin );
    cout<<"enter roll no : ";
    cin>> t.rollno ;
    fflush(stdin) ;
  }
```

```
   int total_marks(total& t)
   {
    t.total_marks = t.T1 + t.T2 + t.T3 + t.P1 + t.P2 ;
    return t.total_marks ;
    }

    char grade(total & t)
    {
      if(t.total_marks /350 > 0.8)
       t.grade = 'A' ;
       else if ((t.total_marks/350) < 0.8 &&  (t.total_marks/100)>0.7)
       t.grade = 'B' ;
       else if ((t.total_marks/350) < 0.7 &&  (t.total_marks/100)>0.6)
       t.grade = 'C' ;
        else if ((t.total_marks/350) < 0.6 &&  (t.total_marks/100)>0.5)
       t.grade = 'D' ;
       else
       t.grade = 'F' ;

       return t.grade ;
    }
  };
```

```
int main()
{   administration a;
    facality f ;
    cout<<"enter the number of students :";
    int num ;
    fflush(stdin) ;
     cin>> num ;
   total t[num] ;
   for(int i =0 ; i<num ;i++)
   {
     a.get_student(t[i]) ;
     f.get_exam_marks (t[i])  ;
     a.total_marks(t[i]) ;
     a.grade(t[i]) ;
   }
      cout<<"\n\n    print name and grade of students in ascending order of grade .if grades are
then print name in respective alphabatical order .\n";
      cout<<"     using function 'sorting1 ' \n\n" ;
      f.sorting2(&t[0],num) ;
      cout << " \n\n print name and grade in alphabatical order of name of student\n      using
function 'sorting2' \n\n";
      f.sorting1(&t[0],num) ;
       cout<<"\n\nserch by first name of student\n     using function 'searching()'\n\n " ;
      cout<<"\n\nenter the first name to search : " ;
      string s ;
      cin >> s ;
    fflush(stdin) ;
    f.searching(&t[0],num , s) ;

    return 0 ;
    }
```

Q6:
Code:

```
#include <iostream>
#include <string>
#include <vector>
#include <stdio.h>
#include <string.h>
using namespace std;
class Landline
{
private:
    string subscriber_name;
    long long int subscriber_number;

public:
    Landline();
    Landline(string, long long int);
    void call(long long int);
    void receive();
};
Landline::Landline()
{
    subscriber_name = "null";
    subscriber_number = 1234567890;
}
Landline::Landline(string subscriber_name_in, long long int subscriber_number_in)
{
    subscriber_name = subscriber_name_in;
    subscriber_number = subscriber_number_in;
}
void Landline::call(long long int number_in)
{
    cout << "Calling to number: " << number_in << " ..." << endl;
}
void Landline::receive()
{
    cout << "Receiving call ...." << endl;
}

class Mobile : public Landline
{
private:
    string subscriber_name;
    long long int subscriber_number;
    vector<pair<string, long long int> > Phonebook;
    long long int dailed_numbers[20];
    int dailed_count;

public:
    void call_by_name();
    long long int getNumberForName(string);
    void init_Phonebook(vector<pair<string, long long int> >);
    void update_dailed_numbers();
    void call_from_dailed_numbers();
    Mobile();
    Mobile(string, long long int);
};
Mobile::Mobile()
{
    subscriber_name = "MobileUserName";
    subscriber_number = 1234567890;
    dailed_count = 0;
}
Mobile::Mobile(string Mob_sub_name, long long int mob_sub_number)
{
    subscriber_name = Mob_sub_name;
    subscriber_number = mob_sub_number;
    dailed_count = 0;
}
void Mobile::call_by_name()
{
    cout << "Printing PhoneBook contacts ...." << endl;
    for (int i = 0; i < Phonebook.size(); i++)
    {
        cout << Phonebook[i].first << " "
             << Phonebook[i].second << endl;
    }
    string name_in;
    cout << "\nEnter Name from above list to call : ";
    cin >> name_in;
    long long int number_in = 0;
    for (int i = 0; i < Phonebook.size(); i++)
    {
        if (Phonebook[i].first == name_in)
            number_in = Phonebook[i].second;
    }
    if (number_in != 0)
    Landline:
        call(number_in);
    else
        cout << "name is not there in phonebook !" << endl;
}
long long int Mobile::getNumberForName(string name_in)
{
    long long int num = 0;
    return num;
}
void Mobile::init_Phonebook(vector<pair<string, long long int> > vec_in)
{
    Phonebook = vec_in;
}
void Mobile::update_dailed_numbers()
{
}
void Mobile::call_from_dailed_numbers()
{
    long long int number_new_in;
    int index_in;
    if (dailed_count == 0)
    {
        cout << "there are no dailed numbers.." << endl;
        cout << "Enter number you want to call: ";
        cin >> number_new_in;
    }
    else
    {
        cout << " list of Dailed numbers : " << endl;
        int i = 0;
        for (i = 0; i < dailed_count; i++)
        {
            cout << i + 1 << ") " << dailed_numbers[i] << endl;
        }
        cout << "Enter your choice : ";
        cin >> index_in;
        number_new_in = dailed_numbers[index_in - 1];
    }
Landline:
    call(number_new_in);
    if (dailed_count < 20)
    {
        dailed_numbers[dailed_count] = number_new_in;
        dailed_count++;
    }
    else if (dailed_count = 20)
    {
        dailed_numbers[0] = dailed_numbers[index_in - 1];
    }
}
int main()
{
    vector<pair<string, long long int> > Phonebook_vector;
    string names[] = {"name1", "name2", "name3", "name4", "name5", "name6", "name7",
"name8", "name9", "name10", "name11", "name12", "name13", "name14", "name15", "name16",
"name17", "name18", "name19", "name20"};
    long long int numbers[] = {9999999991, 9999999992, 9999999993, 9999999994, 9999999995,
9999999996, 9999999997, 9999999998, 9999999999, 9999999910, 9999999911, 9999999912,
9999999913, 9999999914, 9999999915, 9999999916, 9999999917, 9999999918, 9999999919,
9999999920};

    int n = sizeof(names) / sizeof(names[0]);

    for (int i = 0; i < n; i++)
        Phonebook_vector.push_back(make_pair(names[i], numbers[i]));

    Mobile MobileObject("IronMan", 1234567890);
    MobileObject.init_Phonebook(Phonebook_vector);

    MobileObject.call_by_name();
    MobileObject.call_from_dailed_numbers();
    MobileObject.call_from_dailed_numbers();

    return 0;
}
```