Problem 1. Write C++ code in object oriented approach for the students. Student can be rewarded from the department if he gets good GPA in a semester and solved at least 100 ACM problems in the last year. Department will publish the top 5 students name in their honor board. As a student of CSE, write OOP code for the project.

Source code:

```
#include<bits/stdc++.h>
using namespace std;
float array[10];

class Person
{
public:
   string Name;
   int Id;
 Person(string name, int id);
};
Person::Person(string name, int id)
{
   Name=name;
   Id=id;
}
class Student:public Person
{
private:
   float GPA;
   int ACM_Solve;

public:
          Student(string n, int a, float g, int s) : Person(n,a), GPA(g), ACM_Solve(s) {}
          void showStudent();

          int checkGPA(float gpa)
   {
      if(ACM_Solve>=100 && GPA>=3.2 )
         return 1;

      else return 0;
   }
float accessGPA()
   {
      return GPA;
   }
};

void Student::showStudent()
{
   cout << setw(10) << Name << setw(13) << Id << setw(10) <<  GPA << setw(13) << ACM_Solve << endl <<
endl;
}

int main()
{
   int i,j;
   Student *performance[10]=
   {
      new Student("Maria",1001,3.40,45),
      new Student("Shara",1002,3.91,110),
      new Student("Jannat",1003,3.45,150),
      new Student("Fariha",1004,3.26,310),
```

```
        new Student("Rahim",1005,3.28,250),
        new Student("Abir",1006,3.88,100),
        new Student("Jems",1007,3.63,200),
        new Student("Atif",1008,3.27,99)
    };

    float temp;
    int counter=0;
    float maximum;
   for(i=7;i>=0;i--)
    {
        array[i]=performance[i]->accessGPA();
        for(j=0;j<7-i;j++)
        {
            if(array[j]>array[j+1])
            {
                temp=array[j+1];
                array[j+1]=array[j];
                array[j] = temp ;
            }
        }
    }

    cout << "TOP 5 students on the Honor Board are:" << endl << endl;
    cout << setw(10) << "Student Name"  <<  setw(10) << "ID"  << setw(10) << "GPA" << setw(25) << "ACM
Problems Solve" << endl  ;
    cout << setw(10) << "_____________"  <<  setw(10) << "__"  << setw(10) << "___" << setw(25) <<
"_____________________" << endl << endl ;

    for(i=7;i>=0;i--)
    {
        maximum=array[i];

        for(j=0;j<7;j++)
        {
            if(performance[j]->checkGPA(maximum)==1)
            {
                counter++;
                performance[j]->showStudent();
            }

            if(counter==5) break;
        }
    }
    return 0;
}
```

Output:

```
TOP 5 students on the Honor Board are:

Student Name        ID       GPA       ACM Problems Solve
____________        __       ___       __________________

     Shara        1002      3.91              110
    Jannat        1003      3.45              150
    Fariha        1004      3.26              310
     Rahim        1005      3.28              250
      Abir        1006      3.88              100

Process returned 0 (0x0)   execution time : 0.090 s
Press any key to continue.
```

Problem 2. Like Uber, CNG owners want to develop a software which can communicate with customer and driver. A customer needs to say his/her destination and nearer CNG driver will get message from the Uber company with name and mobile number of the customer. The driver can make a direct call and if the customer confirms the ride then driver will come to spot and carry the passenger. As a student of CSE, write OOP code for the project. You need to pass message from one end to another end to make communications.

Source Code:

```
#include<iostream>
#include<math.h>
#include<fstream>                    // read and write on file.
using namespace std;
class Person
{
protected:
   string name;
   int n_id;
public:
   Person(){}
   Person(string name,int n_id)
   {
      this->name=name;
      this->n_id=n_id;
   }
};
class Driver: public Person
{
   int mobile_number;
   string current_location;

public:
   double lat;
   double lng;
  Driver (){};
   Driver(string name,int n_id,int mobile_number,string current_location,double lat,double
lng):Person(name,n_id)
   {
      this->mobile_number=mobile_number;
     this-> current_location=current_location;
     this-> lat=lat;
     this-> lng=lng;
   }
    void driverfile ()
   {
      ofstream myfile2; /// Write in text file same folder.
       myfile2.open ("Driver Information.txt");
      myfile2<<"\nDriver Details "<<endl;
      myfile2<<"------------------------------"<<endl;
      myfile2<<"Mobile number :"<<mobile_number<<endl;
      myfile2<<"current location   : "<<current_location<<endl;
      myfile2<<"Latitude :"<<lat<<"  degree "<<endl;
      myfile2<<"Longitude :  "<<lng<<" degree "<<endl;
      myfile2<<"Name          :"<<name<<endl;
      myfile2<<"National id   :"<<n_id<<endl;
      myfile2<<"--------------------------------";
      myfile2.close();
   }
   void booking ()
   {
      cout<<"a seat has been confirmed"<<endl;
```

```
    }
};
class Uber
{
    Driver driver1,driver2;
public:
    void driver_list(Driver &driver1,Driver &driver2)
    {
      this-> driver1=driver1;
      this-> driver2=driver2;
    }
    double calculate_distance( double customer_lat, double customer_lng, double driver_lat, double driver_lng )
{
    int nRadius = 6371;

    double nDLat = (driver_lat - customer_lat) * (M_PI/180);
    double nDLon = (driver_lng-customer_lng) * (M_PI/180);

        double nA = pow ( sin(nDLat/2), 2 ) + cos(customer_lat) * cos(driver_lat) * pow ( sin(nDLon/2), 2 );

    double nC = 2 * atan2( sqrt(nA), sqrt( 1 - nA ));
    double distance = nRadius * nC;
    return distance;
}
    void contact_with_driver(double customer_lat,double customer_lng)
    {
       double distancefromdriver1,distancefromdriver2;

     distancefromdriver1=calculate_distance(customer_lat,customer_lng,driver1.lat,driver1.lng);
     distancefromdriver2=calculate_distance(customer_lat,customer_lng,driver2.lat,driver2.lng);

     if(distancefromdriver1<distancefromdriver2)
        driver1.booking();
     else
        driver2.booking();
    }
};
class Customer:public Person
{
    int mobile_number;
    string destination;
    double lat;
    double lng;
public:
    Customer(){}
    Customer(string name,int n_id,int mobile_number,string destination,double lat,double lng):Person(name,n_id)
    {
      this-> mobile_number=mobile_number;
       this->destination=destination;
      this-> lat=lat;
      this-> lng=lng;
    }
       friend class Uber;
    void customerfile ()
    {
       ofstream myfile1; /// Write in text file in same folder.
    myfile1.open ("Customer Information.txt");
    myfile1<<"\nCustomer Details "<<endl;
    myfile1<<"----------------------------"<<endl;
    myfile1<<"Mobile number :"<<mobile_number<<endl;
```

```
    myfile1<<"Latitude :"<<lat<<"  degree "<<endl;
    myfile1<<"Longitude:  "<<lng<<" degree "<<endl;
    myfile1<<"Destination   :  "<<destination<<endl;
    myfile1<<"Name         :"<<name<<endl;
    myfile1<<"National id   :"<<n_id<<endl;
    myfile1<<"----------------------------"<<endl;
    }
    void contact_with_uber(Uber &uber)
    {
       double lat,lng;
      this-> lat=lat;
      this-> lng=lng;
     cout<<"\n\nFor customer "<<this->name<<", whose Destination is  "<<this->destination<<", "; ///in console.
      uber .contact_with_driver(lat,lng);
    }
};
int main()
{
    Customer customer("Abir",1,2345,"Comilla",23,91);

    Driver driver1("Asif",2,72567,"Comilla",21,93);
    Driver driver2("Arif",3,8652,"Feni",22,91);

    driver1.driverfile();
    customer.customerfile();

    Uber uber;
    uber.driver_list(driver1,driver2);
    customer.contact_with_uber(uber);
}
```

Output:

```
For customer Abir, whose Destination is  Comilla, a seat has been confirmed

Process returned 0 (0x0)   execution time : 0.110 s
Press any key to continue.
```

File 1 Data:

Customer Details

Mobile number :2345
Latitude :23 degree
Longitude: 91 degree
Destination   : Comilla
Name   :Abir
National id   :1

File 1 Data:

Driver Details

Mobile number :72567
current location   : Comilla
Latitude :21 degree
Longitude : 93 degree
Name   :Asif
National id   :2

Problem: 3. CSE department wants to manage a Fast-food shop in its premise. Every student can make pre-order of his/her breakfast before 10 PM of the previous day .A sales person can manage the data and sells the preordered item to the students .If a student pre-ordered before but not take his/her breakfast and the sales person can inform it to the department. If he will not be illegible to pre-order the breakfast another time. As a student of CSE, write OOP code for the project .All the communications will be held by message.

Source code:

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

class Date
{
   public:
   string date;
   Date(){};
   Date(string date)
   {
      this->date=date;
   }
};
class Person
{
public:
   string name;
   Date date1;
   Person(){};

   Person(string name)
   {
     this->name=name;
   }
   Person(string name,string date)
   {
      this->name=name;
      date1.date=date;
   };

};

class Department
{
public:
   string dpt_name;
   Department(){};
   Department(string name)
   {
      dpt_name=name;
   }
};
class Student : public Person ///student function of person class
{
  public:
     string id;
     Department department;
     Student(){};
     Student(string name,string date,Department dpt_name,string id_no) : Person(name,date)
```

```
    {
      id=id_no;
      department=dpt_name;
    }
    view_student_info()
    {
       cout<<"Student name: "<<name<<endl;
       cout<<"Department: "<<department.dpt_name<<endl;
       cout<<"Student ID: "<<id<<endl;
       cout<<"Students Birthday: "<<date1.date<<endl;
    }
    friend orderfood();
    friend blacklist();
};

class Sellsman : public Person ///sellsman function of person class
{
public:
   int seller_no;
   Sellsman(){};
   Sellsman(string name,string date2,int seller_no): Person(name,date2)
   {
     this->seller_no=seller_no;
   };
   view_sellsman()
   {
      cout<<"Sells person Name: "<<name<<endl;
      cout<<"Birthday: "<<date1.date<<endl;
   }
   friend orderfood();

};
class Login
{
   string user_name;
   string password;
public:
   Login(){};
   Login(string user_name,string password)
   {
     this->user_name=user_name;
      this->password=password;
   }
   friend blacklist();
};

class Faculty : public Person
{
public:
   string designation;
   Department department;
   Login login;
   Faculty(){};
   Faculty(string designation,string name,Department department1,Login *login2):Person(name)
   {
     this->designation=designation;
     department=department1;
     login=*login2;
   }
   view_faculty()
```

```
    {
       cout<<"Name: "<<name<<endl;
       cout<<"Department: "<<department.dpt_name<<endl;
       cout<<"Designation: "<<designation<<endl;
    }
    Blacklist(int order_data[],string deliver_data[],Student *student[],string blacklist[])
    {
       cout<<"Blacklisted names are: \n";
       for(int n=0;n<4;n++)
       {
          if(order_data[n]==1)
         if(deliver_data[n]=="Didn't")
         {
           blacklist[n]=student[n]->name;
           cout<<"Name: "<<student[n]->name<<"\nID: "<<student[n]->id<<"\nYou have been blacklisted,
Contact with Department"<<endl;
         }
       }
    };
};
    int view_system_time()
    {
    time_t k = time(0);
    struct tm *t = localtime(&k);
    cout<< t->tm_hour << ":" << t->tm_min << endl;
    return t->tm_hour;
    }

int orderfood(Student *student)
{
    int tm=view_system_time();
    if(tm>22)
    {
       cout<<"Sorry, You cannot Place the pre-order.\n";
       return -1;
    }
    else
    {
       cout<<"Name: "<<student->name<<"\nID: "<<student->id<<"\n Your Order Has been placed.\n";
       return 1;
    }
}
int main()
{
    int number_of_students=4;
    int temp=number_of_students;
    Department department("CSE");
    Student *students[number_of_students]=
    {
       new Student("Abir","20-6-1996",department,"1001"),
       new Student("Atif","13-3-1998",department,"1002"),
       new Student("Araf","21-11-1996",department,"1003"),
       new Student("jannat","05-07-1997",department,"1004")
    };

    Sellsman sells_person("Sahin","13-03-1985",1);
    sells_person.view_sellsman();
    Login *login_data[2]={
              new Login("Amir","111"),
              new Login("Tamim","222")
```

```
            };
    Faculty *faculty_member[2]=
    {
       new Faculty("Professor","Rafikul Islam",department,login_data[0]),
       new Faculty("Lecturer","Rabiul Islam",department,login_data[1])
    };
    for(int i=0;i<3;i++)
       students[i]->view_student_info();

    for(int i=0;i<2;i++)
    faculty_member[i]->view_faculty();

    int order_data[number_of_students],g=0;
    for(int i=0;i<number_of_students;i++)
    {
      g=orderfood(students[i]);
       order_data[i]=g;

    }
    string delivery_info[number_of_students]={"ordered","Didn't","ordered","Didn't"};
     string blacklisted_members[number_of_students];
     faculty_member[0]->Blacklist(order_data,delivery_info,students,blacklisted_members);

     ofstream students_file;
     students_file.open("Students.csv");
     students_file<<"Students are: \n";
     students_file<<"Roll,Name,Department\n";
     for(int i=0;i<number_of_students;i++)
     {
       students_file<<students[i]->id<<","<<students[i]->name<<","<<students[i]->department.dpt_name<<endl;
     }
     ofstream blacklist_file;                    ///Blacklist part
     blacklist_file.open("Blacklist.csv");

     int counter=1;
      blacklist_file<<"Blacklisted names are:\n";
      blacklist_file<<"Roll,Name,Department\n";
      string blacklisted_id[number_of_students];
     for(int i=0;i<number_of_students;i++)
     {
        if(blacklisted_members[i]!="\0")
        {
         blacklist_file<<students[i]->id<<","<<blacklisted_members[i]<<","<<students[i]-
>department.dpt_name<<endl;
           counter++;
           blacklisted_id[i]=students[i]->id;
        }
     }
     blacklist_file.close();
    cout<<"Blacklisted person included in file.\n";
    cout<<"New to order service?\nPress 1 to register or 0 to exit"<<endl;
    int new_order;
    cin>>new_order;
    while(new_order!=0)
    {
       string name,department_name,birthday,roll;
       cout<<"Enter Name: ";
       cin>>name;
       cout<<"\nEnter ID:";
       cin>>roll;
```

```
        cout<<"\nEnter Department: ";
        cin>>department_name;
        cout<<"\nEnter birthday:";
        cin>>birthday;

        students[number_of_students]={new Student(name,birthday,department_name,roll)};
        number_of_students+=1;
        int order_data1;
       order_data1=orderfood(students[number_of_students-1]);
       cout<<"New to order service?\nPress 1 to register or 0 to exit"<<endl;
       cin>>new_order;
       if(new_order!=1)
        break;
    }
  string delivery_data2[number_of_students]={"ordered","Didn't","ordered","Didn't","ordered"};

    for(int i=temp;i<number_of_students;i++)
    {
       students_file<<students[i]->id<<","<<students[i]->name<<","<<students[i]->department.dpt_name<<endl;
    }
    for(int i=0;i<number_of_students;i++)
    {
       if(blacklisted_id[i]!="\0" && students[i]->id==blacklisted_id[i] )
 cout<<students[i]->name<<"\nYou cannot order,You have been blacklisted.\nContact with  Department"<<endl;
       else
          orderfood(students[i]);
    }
}
```

Output:

	A	B	C
1	Students are:		
2	Roll	Name	Department
3	1001	Abir	CSE
4	1002	Atif	CSE
5	1003	Araf	CSE
6	1004	jannat	CSE
7			

A1

	A	B	C
1	Blacklisted	names are:	
2	Roll	Name	Department
3	1002	Atif	CSE
4	1004	jannat	CSE

```
Department: CSE
Designation: Lecturer
22:7
Name: Abir
ID: 1001
 Your Order Has been placed.
22:7
Name: Atif
ID: 1002
 Your Order Has been placed.
22:7
Name: Araf
ID: 1003
 Your Order Has been placed.
22:7
Name: jannat
ID: 1004
 Your Order Has been placed.
Blacklisted names are:
Name: Atif
ID: 1002
You have been blacklisted, Contact with Department
Name: jannat
ID: 1004
You have been blacklisted, Contact with Department
Blacklisted person included in file.
New to order service?
Press 1 to register or 0 to exit
```

Problem No 4: CSE department wants to select Programming Coach for its students. Any student of the department can be a coach. He need to have high profileat least three ACM regional contest participation and number of problem solutions of ACM need to more 300. Students need to apply in the department,if anyone's performance is below the requirement he will discard automatically.As a student of CSE, write OOP code for the project. All the communications willbe held by message.

Source code:

```
#include<bits/stdc++.h>
using namespace std;
class Date
{
public:
   int day,month,year;
   Date() {};
   Date(int d,int m,int y)
   {
      day=d;
      month=m;
      year=y;
   }
};
class Person               ///Person class
{
public:
   string name;
   Date d1;
   Person() {}; ///Constructor-1 /Default Constructor
   Person(string n)
   {
      name=n;
   }
   Person(string n,Date *d)
   {
      name=n;
      d1=*d;
   }
};

   class Department
   {
   public:
      string dept_name;
      Department() {};
      Department(string n)
      {
         dept_name=n;
      }
   };
  class Student : public Person          /// Student class
   {
   public:
      int id,ACM,prob_solve;
      Department d;
      Student() {};                      ///Constructor-1 /Default Constructor
      Student(string n,Date *d2,Department *dp,int dd,int acm,int solve) : Person(n,d2)  ///constructor-2
      {
         id=dd;
         d=*dp;
         ACM=acm;
         prob_solve=solve;
      }
```

```
    View_Student_Info()                    ///function for viewing student info
    {
        cout<<"Student name: "<<name<<endl;
        cout<<"Department: "<<d.dept_name<<endl;
        cout<<"Students Birthday: "<<d1.day<<"-"<<d1.month<<"-"<<d1.year<<endl;
        cout<<"NUMBER OF ACM PARTICIPATION: "<<ACM<<endl;
        cout<<"NUMBER OF SOLVED PROBLEMS: "<<prob_solve<<endl;
    }
};

class Faculty : public Person                          ///faculty class
{
public:
    Student *s;
    string designation;
    Department dp;
    Faculty() {};                                      ///Constructor-1 /Default Constructor
    Faculty(string d,string n,Department *dp1):Person(d)        ///constructor-2
    {
        designation=n;
        dp=*dp1;
    }
    View_Faculty()                                     ///faculty viewing function
    {
        cout<<"Name: "<<name<<endl;
        cout<<"Department: "<<dp.dept_name<<endl;
        cout<<"Designation: "<<designation<<endl;
    }
    int select_coach(Student *s1)
    {
        s=s1;
        if(s->ACM>=3&&s->prob_solve>300)
            return 1;
        else
            return 0;
    }
};
int main()
{
    Department *dp[5]=                 ///department type object declaration
    {
        new Department("CSE"),
        new Department("CSE"),
        new Department("CSE"),
        new Department("CSE"),
        new Department("CSE"),
    };
    Date *dt[4]=                       ///date type object declaration
    {
        new Date(20,8,1996),
        new Date(15,05,1995),
        new Date(13,11,1993),
        new Date(18,02,1983)
    };
    Student *s[3]=                     ///Student type object declaration
    {
        new Student("Abir",dt[0],dp[0],1001,4,301),
        new Student("Atif",dt[1],dp[1],1002,4,400),
        new Student("Araf",dt[2],dp[2],1003,2,200)
    };
```

```
    Faculty *f[2]=                          ///faculty object declaration.
    {
       new Faculty("Fahim Ahmed","Professor",dp[3]),
       new Faculty("Jahid Ahmed","Lecturer",dp[4])
    };
    cout<<"THE DETAILS OF FACULTY MEMBER: "<<endl<<endl;
    for(int i=0; i<2; i++){
       f[i]->View_Faculty();
       cout<<endl;
    }
    cout<<endl<<endl;
    cout<<"THE DETAILS OF STUDENTS: "<<endl<<endl;
    for(int i=0; i<3; i++){
       s[i]->View_Student_Info();
       cout<<endl;
    }
    int x;
    cout<<endl<<endl;
    for(int i=0;i<3;i++)
    {
      x=f[i]->select_coach(s[i]);
      if(x==1)
      {
         cout<<"THE STUDENT CAN BE SELECTED AS COACH.."<<endl;
         cout<<"THE DETAILS OF THE STUDENT IS: "<<endl;
         s[i]->View_Student_Info();
         cout<<endl<<endl;
      }
      else
      {
         cout<<"THE STUDENT IS NOT SELECTED AS COACH.."<<endl;
         cout<<"THE DETAILS OF THE STUDENT IS: "<<endl;
         s[i]->View_Student_Info();
         cout<<endl<<endl;
      }
    }
    return 0;
}
```

Output:

```
THE DETAILS OF FACULTY MEMBER:

Name: Fahim Ahmed
Department: CSE
Designation: Professor

Name: Jahid Ahmed
Department: CSE
Designation: Lecturer



THE DETAILS OF STUDENTS:

Student name: Abir
Department: CSE
Students Birthday: 20-8-1996
NUMBER OF ACM PARTICIPATION: 4
NUMBER OF SOLVED PROBLEMS: 301

Student name: Atif
Department: CSE
Students Birthday: 15-5-1995
NUMBER OF ACM PARTICIPATION: 4
NUMBER OF SOLVED PROBLEMS: 400

Student name: Araf
Department: CSE
Students Birthday: 13-11-1993
NUMBER OF ACM PARTICIPATION: 2
NUMBER OF SOLVED PROBLEMS: 200



THE STUDENT CAN BE SELECTED AS COACH..
THE DETAILS OF THE STUDENT IS:
Student name: Abir
Department: CSE
Students Birthday: 20-8-1996
NUMBER OF ACM PARTICIPATION: 4
NUMBER OF SOLVED PROBLEMS: 301


THE STUDENT CAN BE SELECTED AS COACH..
THE DETAILS OF THE STUDENT IS:
Student name: Atif
Department: CSE
Students Birthday: 15-5-1995
NUMBER OF ACM PARTICIPATION: 4
NUMBER OF SOLVED PROBLEMS: 400


THE STUDENT IS NOT SELECTED AS COACH..
THE DETAILS OF THE STUDENT IS:
Student name: Araf
Department: CSE
Students Birthday: 13-11-1993
NUMBER OF ACM PARTICIPATION: 2
NUMBER OF SOLVED PROBLEMS: 200



Process returned 0 (0x0)   execution time : 0.170 s
Press any key to continue.
```

Problem no 5: A Mess owner wants to develop a software for its mess members. Everyday mess member meal details will be entered in the software and after the month it will show the bill of the mess member. Mess member deposit at least 1000 tk at beginning of the month. As a student of CSE, write OOP code for the project. All the communications will be held by message.

Source code:

```
#include<bits/stdc++.h>
#include<string>
#include <fstream>

using namespace std;

class Person                    ///Person class
{
public:
   Person() {}                  ///Default Constructor
   string name, date_of_birth;
};

class Owner                     ///Owner Class
{
public:
   class Member : public Person        ///Member class inherits Person class
   {
   public:
      void getName(string names, string d_o_birth)
      {
         name = names;
         date_of_birth = d_o_birth;

      }

      class Meal                    ///Meal class
      {
      public:
         static int counter1,counter2;
         float deposit, total_cost, meal_rate;
         int t_meal;
         string cmeal[15],  date[31];

         get_meal(int total_meal, float deposits, float meal_rate)       /// function for Meal information
         {
            total_cost = total_meal * meal_rate;
            deposit = deposits;
            t_meal = total_meal;
            this->meal_rate=meal_rate;
         }

         meal_count(string dates,string meals,int Members)                  ///function for meal counting
         {
            date[counter1] = dates;
            cmeal[counter1] = meals;
            if(counter2 % Members == Members - 1) counter1++;
            counter2++;

         }
      };


      view(int counter,int DayCount)                              /// function for viewing Meal information
```

```
        {
            cout << "Member No :"<< counter << endl;
            cout << "Name : " << name <<endl;
            cout << "Date of Birth : " << date_of_birth <<endl;
            cout << "----------------------------" <<endl;
            cout<< "Total Deposit --> " << ml.deposit << endl;
            cout << "Total meal    --> " << ml.t_meal << endl;
            cout << "Meal Rate     --> " << ml.meal_rate << endl;
            cout << "Total Cost    --> " << ml.total_cost <<  endl;
            cout << "Total Balance --> " << ml.deposit - ml.total_cost << endl<<endl;
            cout << "----Meal Information of " << name << "----\n"<<endl;
            for(int j = 1; j<DayCount; j++) cout << ml.date[j] << " --> " << ml.cmeal[j] <<endl;
            cout << endl<<endl;
            cout << "----------------------------" <<endl;
            cout << "----------------------------" <<endl<<endl<<endl;
        } Meal ml ;
    };
    Member members[15];
};

view_print(int deposite,int shopping,int TotalMeal,float meal_rate,int Members)
{
    cout << endl << "Meal Informations : " << endl;
    cout << "---------------------------------------" << endl;
    cout << "---------------------------------------" << endl;
    cout << "Total Member : " << Members << endl;
    cout << "Total Deposites : "  << deposite << endl;
    cout << "Total Cost : "  << shopping << endl;
    cout << "Total Meal : "  << TotalMeal << endl;
    cout << "Meal Rate : "  << meal_rate << endl;
    cout << "---------------------------------------" << endl;
    cout << "---------------------------------------" << endl << endl;
}
int Owner::Member::Meal :: counter1;                ///Defination of Static Data Member
int Owner::Member::Meal:: counter2;                    ///Defination of Static Data Member
int main()
{
    Owner owner;


    string name,date_of_birth,deposite,date,cmeal[31][15];
    int counter=0,TotalMeal = 0;

    int DayCount = 0;
    string apps;

    string MeaL[31];
    ifstream file,file_2;

    /***----------------Open csv file for Total Member Counting----------------***/

    file.open("E:/Arman_Name.csv");
    int MemberCount = 0;

    while(file.good())
    {
        getline(file,name,',');
        getline(file,date_of_birth, ',');
        getline(file,deposite, '\n');
        MemberCount++;
```

```
}
MemberCount = MemberCount - 1;

file.close();

/***----------Open csv file for Total Day Counting----------***/
file_2.open("E:/Arman_Meal.csv");       ///Open a csv file named meal

while(file_2.good())                    ///days count (end of the file)
{
   getline(file_2,date,',');

   for(int mem=0; mem<MemberCount; mem++)
   {
      if(mem==MemberCount - 1) getline(file_2, cmeal[mem][counter], '\n'); ///for last person
      else  getline(file_2, cmeal[mem][counter], ',');
   }     ///others
   DayCount++;
}
DayCount = DayCount - 1;

file_2.close();
int total[DayCount]= {0};

/***----------Open csv file for Total Meal Counting----------***/

file_2.open("E:/Arman_Meal.csv");       ///Open a csv file named meal

/// while(file_2.good())                ///days count (end of the file)
for(int i = 0; i<DayCount; i++)
{
   getline(file_2,date,',');

   for(int mem=0; mem<MemberCount; mem++)
   {
      if(mem==MemberCount - 1) getline(file_2, cmeal[mem][counter], '\n'); ///for person 6
      else  getline(file_2, cmeal[mem][counter], ',');         ///others

      for(int i=0; i<DayCount; i++)
         if(counter == i) MeaL[i] = cmeal[mem][counter];

      int DayTMeal[DayCount];
      int num[DayCount];
      memset(DayTMeal, 0, sizeof(DayTMeal));
      memset(num, 0, sizeof(num));
      for(int i = 0; i<DayCount; i++)
      {
         if(counter != i)
            continue;
         if(counter == i)
         {
            for(int j=0; MeaL[i][j]; j++)
            {
               if(MeaL[i][j]>='0' and MeaL[i][j]<='9')
               {
                  num[i] *= 10;
                  num[i] += (MeaL[i][j]-'0');
               }
               else
```

```
                {
                    DayTMeal[i] += num[i];
                    num[i] = 0;
                }
            }
            DayTMeal[i] += num[i];
        }
        else DayTMeal[i] = 0;
    }

    for(int i=0; i<MemberCount; i++)
    {
        if(mem == i )
        {
            for(int j=0; j<DayCount; j++)          /// individual meal count
                total[i] += DayTMeal[j];
        }
    }

    owner.members[mem].ml.meal_count(date,cmeal[mem][counter],MemberCount);
  }
  counter++;
}

file_2.close();              ///file closed (meal)

ifstream bz;
string baz;
int shopping =0,m;
bz.open("E:/Arman_Shopping.csv");                              ///Open a csv file named bazar

while(bz.good())
{
  getline(bz, baz,'\n');
  stringstream gerk(baz);
  gerk >> m;
  shopping += m;
}
shopping = shopping - m;

for(int i=0; i<MemberCount; i++)
  TotalMeal += total[i];

bz.close();
float meal_rate = shopping / (float)TotalMeal;


file.open("E:/Arman_Name.csv");
///Open a csv file named "name"

int deposites = 0,n;
string Name[MemberCount];
for(int i = 0; i<MemberCount; i++)
{
  float final_dep = 0.0;
  getline(file,name,',');
  getline(file,date_of_birth, ',');
  getline(file,deposite, '\n');

  stringstream geek(deposite);                          ///converts strings into integer
```

```
        geek >> final_dep;
        deposites += final_dep;
        Name[i] = name;
        owner.members[i].getName(name,date_of_birth);
        owner.members[i].ml.get_meal(total[i],final_dep,meal_rate);

    }
    file.close();                       ///file closed (name)
//cout << name << endl;
    string password = "**********";
    string getpassword;

//cin >> getpassword;
    while(true)
    {
        cout << "Please enter your Password : " ;
        cin >> getpassword;
        if(password == getpassword)
        {
            view_print(deposites,shopping,TotalMeal,meal_rate,MemberCount); ///overall meal info
            break;
        }
        else cout << endl << "Your password is incorrect,please try again." << endl << endl;
    }
    string NameUpper[MemberCount],NameLower[MemberCount];
    while(true)
    {

        string person;
        cout << "Enter Member Name : " << endl;
        cin >> person;
        for(int i = 0; i<person.size(); i++)
        {
            if(i == 0) person[i]=toupper(person[i]);
            else person[i]=tolower(person[i]);
        }
        int y;
        //              ///individual meal info show
        for(int i = 0; i<MemberCount; i++)
        {
            if(person == Name[i])
            {
                y = i;
                cout << endl << "Mess Members Information :" << endl << endl;
                owner.members[y].view(y+1,DayCount);
                break;
            }
        }

    }
    return 0;
}
```

Output: