Ronaldo A Amaya

Professor Ali

Computer Science 1428-254

May 2, 2019

# Project #6 Report

## Step 1: Defining what the program is to do.

**Purpose:** Following from the 5 previous projects, this one is keeping everything the same yet the set up in the coding is changed just a little bit. Every functions are the same and a bunch of the coding stays the same yet there will only be a few arrays since we are adding in some structures. The functions each do certain things like one of the functions is made in order to pull all of the InputFile information, another function is to make sure it validates everything from the InputFile, and so on and so forth yet instead of arrays being the one to hold the codes, it will be the structures. The main reason is to make a program with arrays being passed through functions.

**Input:** Structures: struct information, string header, string Name, string ID, string address, string CellPhone, string SSN, int Age, int Year, string course[SIZE], string commend, string warn, struct calculate: double Exam[SIZE][six], char letter_grade[SIZE][SIZE], struct compute: information Info, calculate results, Function prototypes: void Input(ifstream &, struct compute info[], int ), void Validate1(ofstream &,struct compute info[], int), void Validate2(ofstream &,struct compute grade[], int), void Numerical(struct compute grade[], int), void LetterGrade(struct compute grade[], int), void Comments(struct compute info[] ,int), void Report(ofstream &, struct compute output[], int). Normal functions: compute allstudents[NumStudents], manystudents=0, Numstudents, small= 2, six = 6, SIZE = 3, longer=5, p1 = 0.10, p2 = 0.15, p3 = 0.20, p4 = 0.25, p5 = 0.30, A_score=90, HighA=95, B_score=80, C_score=70, D_score=60, testweight, testweight2, testweight3, testweight4, testweight5, fin, fout.

**Process:** Before anything, the user will be ask to put how many students are there. Anything that is not equal to 3 will force the user in a while loop until it's 3. After that all of the functions will begin to do their thing. The first function will get all of the information from the inputFile meanwhile the three next functions will validate everything in order to make sure everything is as written as it should be. If something is wrong, it will tell the user. After

that, another function will make the numerical grade. The following function will get the letter grade with the one after that creating the commend note or warn note before the last function that will print everything in outputFile.

.

**Output:** The output will be blank unless they don't have the right input file, which will pop up an error. Also if the user puts the wrong amount of students, the error will pop up before continuing the program. On the Project5_A04853058_outputFile.txt at least will have the header, name of the student, Student ID, their address, their telephone number, their social security number, Student's age, their number of years at texas state, then it will begin the middle loop, being the courses and the inner loop being the five exams. After looping the tests, it will create the average of the tests combined, the average number, the letter grade, and the note (depending on what the person got in the specific course) will be in the middle loop after the inner loop to print into the Output File. After that it will loop until the last student.

## Step 4: Algorithm.

If you don't have the right InputFile:
Display: "Could not open file. Terminating program."

Display: "How many Students?"
Input: NumStudents

If you put the number of students is not 3, there will be a while loop
Until you put the right amount of students, also it will do this:
Display: "Wrong amount of students. Try again. "
Input: NumStudents

If you have the right InputFile:

Open: Project5_A04853058_inputFile.txt as fin source code.
Open: Project5_A04853058_outputFile.txt as fout source code.
//MASSIVE NOTE: This next part will be the functions, each one doing their own thing.
Main function:
Compute: compute allstudents[NumStudents]

Structure : For (manystudents=0; manystudents < NumStudents; manystudents++)

//This loop is to help repeat the process for the students.  All of their
//information and all of the next loop.

Compute: Input(fin, allstudents, manystudents);
Compute: Validate1(fout, allstudents, manystudents);
Compute: Validate2(fout, allstudents, manystudents);
Compute: Numerical(allstudents, manystudents);
Compute: LetterGrade(allstudents, manystudents);
Compute: Comments(allstudents, manystudents);
Compute: Report(fout,allstudents,manystudents);
fin.close();
fout.close();
return 0; //End of program

Input function:

getline(fin,info[manystudents].Info.header);
getline(fin,info[manystudents].Info.Name);
getline(fin,info[manystudents].Info.ID);
getline(fin,info[manystudents].Info.address);
getline(fin,info[manystudents].Info.CellPhone);
getline(fin,info[manystudents].Info.SSN);

Structure 1:  for(int courses= 0; courses < SIZE; courses++)
Structure 2:  for(int j = 0 ; j < longer; j++)
With structures 1 and 2, it will read and store all grades 1-5.
fin.ignore();

//The next functions will be validations to make sure things are actually how they are
suppose to be.
Validate1 function:

if true: info[manystudents].Info.header.length() < 2 or
info[manystudents].Info.header.length() > 100
Write into fout: Error: Student  (manystudents+1)  has an invalid header.  The Number of characters in the student's header must be between 2 and 100, including spaces.
return;

if true: (info[manystudents].Info.Name.length() < 2 or info[manystudents].Info.Name.length() > 50

Write into fout: Error: Student (manystudents+1) has an invalid name. The Number of characters in the student's name must be between 2 and 50, including spaces.
return;

if true: info[manystudents].Info.ID.length()!= 9
Write into fout: Error: Students (manystudents +1) has an invalid Student ID. The number of characters in the Student's ID must equal to 9.
return;

if true: info[manystudents].Info.address.length() < 10 or info[manystudents].Info.address.length() > 100
Write into fout: Error: Student (manystudents+1) has an invalid Address. The Number of characters in the student's address must be between 10 and 100, including spaces.
return;

If true: info[manystudents].Info.CellPhone.length() < 1 or info[manystudents].Info.CellPhone.length() > 15
Write into fout:Error: Student (manystudents+1) has an invalid Phone #. The number of characters in the student's phone number must be between 1 and 15, including spaces.
Return;

If true: info[manystudents].Info.SSN.length() < 1 or info[manystudents].Info.SSN.length() > 11
Write into fout: Error: Student (manystudents+1) has an invalid Social Security #. The Number of characters in the student's Social Security # must be between 1 and 11, including spaces.
return;

if true: info[manystudents].Info.Age < 1 or info[manystudents].Info.Age > 90

Write in fout: Error: The age of Student (manystudents + 1) is invalid. The student's age must be in between 1 and 90.
return;

if true: info[manystudents].Info.Year < 1 or info[manystudents].Info.Year > 90

Write into fout: Error: The amount of years going to Texas State for Student (manystudents + 1) is invalid. The student's number of years must be in between 1 and 90.
return;

//Validate2 Function
if true: grade[manystudents].Info.course[0].length() < 6 or grade[manystudents].Info.course[0].length() > 8
Write into fout: Error: Student (manystudents+1) Course number is invalid. The Number of characters in the student's course number must be in between 6 and 8.
return;

if true: grade[manystudents].Info.course[1].length() < 6 or grade[manystudents].Info.course[1].length() > 8
Write into fout: Error: Student (manystudents+1) Course number is invalid. The Number of characters in the student's course number must be in between 6 and 8.
return;

if true:grade[manystudents].Info.course[2].length() < 6 or grade[manystudents].Info.course[2].length() > 8
Write into fout: Error: Student (manystudents+1) Course number is invalid. The Number of characters in the student's course number must be in between 6 and 8.
return;

Structure 3: for(int d = 0; d < SIZE; d++)
Structure 4: for(int scores =0; scores < longer; scores++)
if true: grade[manystudents].results.Exam[d][scores] < 1.00 or grade[manystudents].results.Exam[d][scores]>100.00
Write into fout: Error: Grade (scores+1) for Student (manystudents + 1) in Course number (manystudents + 1) is Invalid. Grades must be between 1.00 and 100.00 .
break;

//Numerical function
Structure 5: for (int d =0; d < SIZE; d++)

Compute: testweight = grade[manystudents].results.Exam[d][0] * p1;
Compute: testweight2 = grade[manystudents].results.Exam[d][1] * p2;

Compute: testweight3 = grade[manystudents].results.Exam[d][2] * p3;
Compute: testweight4 = grade[manystudents].results.Exam[d][3] * p4;
Compute: testweight5 = grade[manystudents].results.Exam[d][4] * p5;
Compute: grade[manystudents].results.Exam[d][5] = testweight + testweight2 + testweight3 + testweight4 + testweight5;

//LetterGrade function
Structure 6: for (int d =0; d < 3; d++)

if true: grade[manystudents].results.Exam[d][5] greater than A_score

Assign grade[manystudents].results.letter_grade[manystudents][d] to: A

else if true: grade[manystudents].results.Exam[d][5] greater than B_score and grade[manystudents].results.Exam[d][5] less than A_score

Assign grade[manystudents].results.letter_grade[manystudents][d] to: B

else if true: grade[manystudents].results.Exam[d][5] greater than C_score and grade[manystudents].results.Exam[d][5] less than B_score

Assign grade[manystudents].results.letter_grade[manystudents][d] to: C

else if true: grade[manystudents].results.Exam[d][5] greater than D_score and grade[manystudents].results.Exam[d][5] less than C_score

Assign grade[manystudents].results.letter_grade[manystudents][d] to: D

else true:
Assign grade[manystudents].results.letter_grade[manystudents][d] to: F

//Comments function
Input: string Notes [small]={"Congradulations, Your performance is Excellent",
"Your grade is too low and needs improvements"};

Structure 7: for (int d =0; d < SIZE; d++)

```
        if true: info[manystudents].results.Exam[d][5] greater than or equal to HighA
      Assign info[manystudents].Info.commend to: Notes[0]
   else if true: info[manystudents].results.Exam[d][5] less than C_score)
      Assign info[manystudents].Info.warn to: Notes[1]
//Report function
Outer loop:
   Write into fout:  output[manystudents].Info.header
   Write into fout:  Name of Student:output[manystudents].Info.Name
   Write into fout:  Student's ID: output[manystudents].Info.ID
   Write into fout:  Address:  output[manystudents].Info.address
   Write into fout:  Telephone Number: output[manystudents].Info.CellPhone
   Write into fout:  Student Soc. Security #:  output[manystudents].Info.SSN
   Write into fout:  Age:  output[manystudents].Info.Age
   Write into fout:  Number of years at Texas State: output[manystudents].Info.Year

Middle loop:
{
Structure 8: for (int d = 0; d < SIZE; d++)
   fout << setw(31) << "Course number:" << setw(13) <<
output[manystudents].Info.course[d] << endl;
Inner loop:
  {
  Structure 9:  for (int c = 0; c < longer; c++)
   Write into fout: Exam  (c+1) :  output[manystudents].results.Exam[d][c]
}
   Write into fout: Numerical Grade in  output[manystudents].Info.course[d]:
grade[manystudents].results.Exam[d][5]
     Write into fout: Letter Grade in output[manystudents].Info.course[d]:
.letter_grade[manystudents][d]
   if true: output[manystudents].results.Exam[d][5] greater than HighA
         Write into fout: Commend Note: output[manystudents].Info.commend
         Write into fout: " "
   else if true: output[manystudents].results.Exam[d][5] less than C_score
         Write into fout: Warning Note: output[manystudents].Info.warn
         Write into fout: " "
   else true:
         Write into fout: " "
```

## Step 5: Source code and printed OI screen.

```
/*
   Amaya Ronaldo
   Project #6
   A04853058
   Due Date: May 2, 2019

*/
// This last project will be using structures, functions, arrays, loops, and what I have learn to make
// to make a program that will print out the students info and their grades.
#include <iostream>
#include <iomanip> // library for setw() and other i/o manipulators
#include <fstream> // library for file input and output
#include <string>

using namespace std;

   const int small= 2, six = 6, SIZE = 3, longer = 5, super = 9;

   int NumStudents; //number of students

   //Structures
   struct information
   {
      string header;
      string Name;
      string ID;
      string address;
      string CellPhone;
      string SSN;
      int Age;
      int Year;
      string course[SIZE];
      string commend;
      string warn;
   };

   struct calculate
   {
   double Exam[SIZE][six];
   char letter_grade[SIZE][SIZE];
   };

   struct compute
   {
      information Info;
      calculate results;
   };
   //function prototype
   void Input(ifstream &, struct compute info[], int );
   void Validate1(ofstream &,struct compute info[], int);
   void Validate2(ofstream &,struct compute grade[], int);
   void Numerical(struct compute grade[], int);
   void LetterGrade(struct compute grade[], int);
```

```
   void Comments(struct compute info[] ,int);
   void Report(ofstream &, struct compute output[], int);

int main()
{

   const int A_score=90, HighA=95, B_score=80, C_score=70,D_score=60;
   double testweight, testweight2, testweight3, testweight4, testweight5;
   int manystudents;

   ifstream fin; //input file
fin.open("Project6_A04853058_inputFile.txt"); //name of input file and opening inputFile
if (!fin) //If there is no file saying "Project2_A04853058_inputFile.txt", there will be an error.
{
   cout << "Could not open file.  Terminating program." << endl;
   return -1; //Terminating program
}


   ofstream fout; //output file
   fout.open("Project6_A04853058_outputFile.txt"); //opening outputFile and outputFile name

   if (!fout)
{
   cout << "ERROR-Output File failed to open." << endl;
   return -1;
}
   cout << "How many Students? ";
   cin >> NumStudents;

   while(NumStudents!=SIZE)
   {
      cout << "Wrong amount of students.  Try again. ";
      cin >> NumStudents;

      break;
   }

compute allstudents[NumStudents];
   for (manystudents=0; manystudents < NumStudents; manystudents++) //Outer loop for students
{
   Input(fin, allstudents, manystudents);
   Validate1(fout, allstudents, manystudents);
   Validate2(fout, allstudents, manystudents);
   Numerical(allstudents, manystudents);
   LetterGrade(allstudents, manystudents);
   Comments(allstudents,manystudents);
   Report(fout,allstudents,manystudents);
   fin.ignore();
}

   fin.close();
   fout.close();
   return 0; //End of program
}

  void Input(ifstream &fin, struct compute info[], int manystudents)
```

```
 {
   getline(fin,info[manystudents].Info.header);
   getline(fin,info[manystudents].Info.Name);
   getline(fin,info[manystudents].Info.ID);
   getline(fin,info[manystudents].Info.address);
   getline(fin,info[manystudents].Info.CellPhone);
   getline(fin,info[manystudents].Info.SSN);
for (int i = 0; i < SIZE; i++)
{
   getline(fin,info[manystudents].Info.course[i]);
}
   fin >> info[manystudents].Info.Age;
   fin >> info[manystudents].Info.Year;
            {
                for(int courses= 0; courses < SIZE; courses++)
                for(int j = 0 ; j < longer; j++)
                {
                fin >> info[manystudents].results.Exam[courses][j];
                fin.ignore();
                }
              }
}


   void Validate1(ofstream &fout,struct compute info[], int manystudents)
   {
   if (info[manystudents].Info.header.length() < 2 || info[manystudents].Info.header.length() > 100)
   {
      fout << "Error: Student " << (manystudents+1) << " has an invalid header." << endl;
      fout << "The Number of characters in the student's header must be between 2 and 100, "
      << "including spaces." << endl;
      return;
   }

   if (info[manystudents].Info.Name.length() < 2 || info[manystudents].Info.Name.length() > 50)
   {
      fout << "Error: Student " << (manystudents+1) << " has an invalid name." << endl;
      fout << "The Number of characters in the student's name must be between 2 and 50, "
      << "including spaces." << endl;
      return;
   }

      if (info[manystudents].Info.ID.length()!= 9)
      {
         fout << "Error: Students " << (manystudents +1) << " has an invalid Student ID." << endl;
         fout << "The number of characters in the Student's ID must equal to 9." << endl;
         return;
      }

      if (info[manystudents].Info.address.length() < 10 || info[manystudents].Info.address.length() > 100)
      {
         fout << "Error: Student " << (manystudents+1) << " has an invalid Address." << endl;
         fout << "The Number of characters in the student's address must be between 10 and 100, "
         << "including spaces." << endl;
         return;
      }
```

```
    if (info[manystudents].Info.CellPhone.length() < 1 || info[manystudents].Info.CellPhone.length() > 15)
    {
    fout << "Error: Student " << (manystudents+1) << "has an invalid Phone #." << endl;
    fout << "The number of characters in the student's phone number must be between 1 and "
    << "15, including spaces." << endl;
    return;
    }


    if (info[manystudents].Info.SSN.length() < 1 || info[manystudents].Info.SSN.length() > 11)
    {
       fout << "Error: Student " << (manystudents+1) << " has an invalid Social Security #." << endl;
       fout << "The Number of characters in the student's Social Security # must be between 1 and 11, "
       << "including spaces." << endl;
       return;
    }
  if (info[manystudents].Info.Age < 1 || info[manystudents].Info.Age > 90)
  {
    fout << "Error: The age of Student " << (manystudents + 1) << " is invalid." << endl;
    fout << "The student's age must be in between 1 and 90. " << endl;
    return;
  }

    if (info[manystudents].Info.Year < 1 || info[manystudents].Info.Year > 90)
  {
    fout << "Error: The amount of years going to Texas State for Student " << (manystudents + 1) << " is invalid." << endl;
    fout << "The student's number of years must be in between 1 and 90. " << endl;
    return;
  }
}

  void Validate2(ofstream &fout,struct compute grade[], int manystudents)
  {
        if (grade[manystudents].Info.course[0].length() < 6 || grade[manystudents].Info.course[0].length() > 8)
    {
       fout << "Error: Student " << (manystudents+1) << " Course number is invalid." << endl;
       fout << "The Number of characters in the student's course number must be in between 6 and 8. "<< endl;
       return;
    }

          if (grade[manystudents].Info.course[1].length() < 6 || grade[manystudents].Info.course[1].length() > 8)
    {
       fout << "Error: Student " << (manystudents+1) << " Course number is invalid." << endl;
       fout << "The Number of characters in the student's course number must be in between 6 and 8. "<< endl;
       return;
    }

          if (grade[manystudents].Info.course[2].length() < 6 || grade[manystudents].Info.course[2].length() > 8)
    {
       fout << "Error: Student " << (manystudents+1) << " Course number is invalid." << endl;
       fout << "The Number of characters in the student's course number must be in between 6 and 8. "<< endl;
       return;
    }

   for(int d = 0; d < SIZE; d++)
   for(int scores =0; scores < longer; scores++)
   if (grade[manystudents].results.Exam[d][scores] < 1.00 || grade[manystudents].results.Exam[d][scores]>100.00)
```

```
   {
      fout << "Error: Grade " << (scores+1) << " for Student " << (manystudents + 1);
      fout << " in Course number " << (manystudents + 1) << " is Invalid." << endl;
      fout << "Grades must be between 1.00 and 100.00 ." << endl;
      break;
   }
}

  void Numerical(struct compute grade[], int manystudents)
{
   double p1 = 0.10, p2 = 0.15, p3 = 0.20, p4 = 0.25, p5 = 0.30;
   double testweight, testweight2, testweight3, testweight4, testweight5;
  for (int d =0; d < SIZE; d++)
   {
            testweight = grade[manystudents].results.Exam[d][0] * p1;
     testweight2 = grade[manystudents].results.Exam[d][1] * p2;
     testweight3 = grade[manystudents].results.Exam[d][2] * p3;
     testweight4 = grade[manystudents].results.Exam[d][3] * p4;
     testweight5 = grade[manystudents].results.Exam[d][4] * p5;
   grade[manystudents].results.Exam[d][5]  = testweight + testweight2 + testweight3 + testweight4 + testweight5;
   }

}

  void LetterGrade(struct compute grade[], int manystudents)
 {
     const int A_score=90, B_score=80, C_score=70,D_score=60;
        for (int d =0; d < 3; d++)
        {
        if ( grade[manystudents].results.Exam[d][5] > A_score)
        {
          grade[manystudents].results.letter_grade[manystudents][d] = 'A';
        }
        else if (grade[manystudents].results.Exam[d][5] > B_score && grade[manystudents].results.Exam[d][5] < A_score)
        {
           grade[manystudents].results.letter_grade[manystudents][d] = 'B';
        }
        else if (grade[manystudents].results.Exam[d][5] > C_score && grade[manystudents].results.Exam[d][5] < B_score)
        {
             grade[manystudents].results.letter_grade[manystudents][d] = 'C';
        }
        else if (grade[manystudents].results.Exam[d][5] > D_score &&  grade[manystudents].results.Exam[d][5] < C_score)
        {
             grade[manystudents].results.letter_grade[manystudents][d] = 'D';
        }
        else
        {
             grade[manystudents].results.letter_grade[manystudents][d] = 'F';
        }
     }
 }


  void Comments(struct compute info[] ,int manystudents)
  {
     const int HighA=95,C_score=70;
     string Notes [small]={"Congradulations, Your performance is Excellent",
```

```
        "Your grade is too low and needs improvements"};

    for (int d =0; d < SIZE; d++)
    {
          if (info[manystudents].results.Exam[d][5] >= HighA) //If grade is equal or higher than 95, they get a commend note
   {
     info[manystudents].Info.commend = Notes[0];
   }
   else if (info[manystudents].results.Exam[d][5]<C_score) // If grade is lower than a 70, they get a warning
   {
      info[manystudents].Info.warn = Notes[1];
   }
    }
   }

   void Report(ofstream &fout, struct compute output[], int manystudents)
{
   const int HighA=95,C_score=70;
   fout << setw(41) << output[manystudents].Info.header << endl;
   fout << setw(31) <<"Name of Student:" << right << setw(20)<< output[manystudents].Info.Name << endl;
   fout << setw(31) << "Student's ID:" << right << setw(16) << output[manystudents].Info.ID << endl;
   fout << setw(31) <<"Address:" << setw(37) << right << output[manystudents].Info.address << endl;
   fout << setw(31) <<"Telephone Number:" << setw(20) << output[manystudents].Info.CellPhone << endl;
   fout << setw(31) <<"Student Soc. Security #:" << setw(18) << output[manystudents].Info.SSN << endl;
   fout << setw(31)<<"Age:" << setw(9) << right << output[manystudents].Info.Age << endl;
   fout << setw(1)<<"Number of years at Texas State:" << setw(8) << right << output[manystudents].Info.Year << endl;

for (int d = 0; d < SIZE; d++) // Since courses is connected to string array, the d is to help separate the courses and their tests.  Also middle loop
{
   fout << setw(31) << "Course number:" << setw(13) <<  output[manystudents].Info.course[d] << endl;

    for (int c = 0; c < longer; c++) //A for loop to print out the tests and numerical grade.  Inner loop
   {
   fout << setprecision(2) << fixed << showpoint << setw(29) << "Exam " << (c+1) << ":" << setw(11) << right <<
output[manystudents].results.Exam[d][c] << endl;
   }
   fout << setprecision(2) << fixed << showpoint << setw(24) << "Numerical Grade in " << output[manystudents].Info.course[d] << ":" <<
setw(12) << output[manystudents].results.Exam[d][5] << endl;
   fout << setw(24) << "Letter Grade in " << output[manystudents].Info.course[d] << ":" << setw(8) <<
output[manystudents].results.letter_grade[manystudents][d] << endl;

      {
   if (output[manystudents].results.Exam[d][5] >= HighA) //If grade is equal or higher than 95, they get a commend note
   {
      fout << setw(24) << "Commend Note: " <<  output[manystudents].Info.commend << endl;
      fout << "\n";
   }
   else if (output[manystudents].results.Exam[d][5] <C_score) // If grade is lower than a 70, they get a warning
   {
      fout << setw(24) << "Warning Note: " <<  output[manystudents].Info.warn << endl;
      fout << "\n";
   }
   else
      fout << "\n";//Space

      }
}
```

}

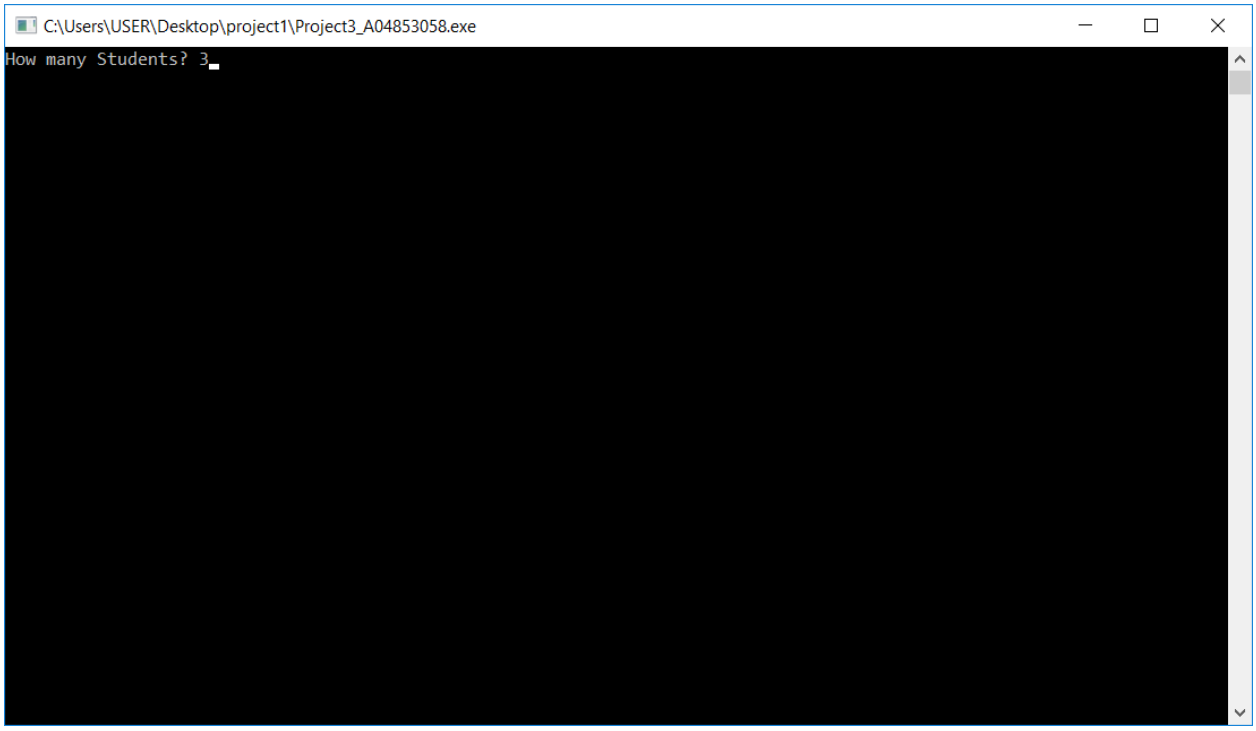
Project6_A04853058_inputFile.txt:

```
Student Grade Sheet, Department of Computer Science, Texas State University
Ronaldo Amaya
A04853058
616 N. Dr.,San Marcos,TX 78666
(210)551-5353
111-22-3333
CS1428
CS2808
CS3358
20
1
95.9
95.2
98.5
90.8
97.7
50.9
55.2
60.5
65.8
45.7
72.5
65.9
91.7
88.8
69.3
Student Grade Sheet, Department of Computer Science, Texas State University
Bergo Bond
A00075388
616 N. Dr.,San Marcos,TX 78666
(551)451-1333
101-01-0000
CS1428
CS2808
CS3358
```

21
3
99.9
91.2
100.0
100.0
95.0
22.8
69.2
98.4
68.2
77.7
88.8
95.5
94.1
51.5
98.2
Student Grade Sheet, Department of Computer Science, Texas State University
Kassandra Knight
A00012458
616 N. Dr.,San Marcos,TX 78666
(210)551-1353
555-52-4533
CS1428
CS2808
CS3358
18
1
99.9
95.2
98.5
97.2
55.5
100.0
40.9
81.0
70.7
52.2
72.3

80.2
99.1
44.4
75.2

Output:




Project6_A04853058_outputFile.txt:

Student Grade Sheet, Department of Computer Science, Texas State University

Name of Student: Ronaldo Amaya

Student's ID: A04853058

Address: 616 N. Dr.,San Marcos,TX 78666

Telephone Number: (210)551-5353

Student Soc. Security #: 111-22-3333

Age: 20

Number of years at Texas State: 1

Course number: CS1428

Exam 1: 95.9

Exam 2: 95.2

Exam 3: 98.5
Exam 4: 90.8
Exam 5: 97.7
Numerical Grade in CS1428: 95.58
Letter Grade in CS1428: A
Commend Note: Congradulations, Your performance is Excellent

Course number: CS2808
Exam 1: 50.90
Exam 2: 55.20
Exam 3: 60.50
Exam 4: 65.80
Exam 5: 45.70
Numerical Grade in CS2808: 55.63
Letter Grade in CS2808: F
Warning Note: Your grade is too low and needs improvements

Course number: CS3358
Exam 1: 72.50
Exam 2: 65.90
Exam 3: 91.70
Exam 4: 88.80
Exam 5: 69.30
Numerical Grade in CS3358: 78.46
Letter Grade in CS3358: C

Student Grade Sheet, Department of Computer Science, Texas State University
Name of Student: Bergo Bond
Student's ID: A00075388
Address: 616 N. Dr.,San Marcos,TX 78666
Telephone Number: (551)451-1333
Student Soc. Security #: 101-01-0000
Age: 21
Number of years at Texas State: 3
Course number: CS1428
Exam 1: 99.90
Exam 2: 91.20
Exam 3: 100.00
Exam 4: 100.00

Exam 5: 95.00
Numerical Grade in CS1428: 97.17
Letter Grade in CS1428: A
Commend Note: Congradulations, Your performance is Excellent

Course number: CS2808
Exam 1: 22.80
Exam 2: 69.20
Exam 3: 98.40
Exam 4: 68.20
Exam 5: 77.70
Numerical Grade in CS2808: 72.70
Letter Grade in CS2808: C

Course number: CS3358
Exam 1: 88.80
Exam 2: 95.50
Exam 3: 94.10
Exam 4: 51.50
Exam 5: 98.20
Numerical Grade in CS3358: 84.36
Letter Grade in CS3358: B

Student Grade Sheet, Department of Computer Science, Texas State University
Name of Student: Kassandra Knight
Student's ID: A00012458
Address: 616 N. Dr.,San Marcos,TX 78666
Telephone Number: (210)551-1353
Student Soc. Security #: 555-52-4533
Age: 18
Number of years at Texas State: 1
Course number: CS1428
Exam 1: 99.90
Exam 2: 95.20
Exam 3: 98.50
Exam 4: 97.20
Exam 5: 55.50
Numerical Grade in CS1428: 84.92
Letter Grade in CS1428: B

Course number: CS2808
Exam 1: 100.00
Exam 2: 40.90
Exam 3: 81.00
Exam 4: 70.70
Exam 5: 52.20
Numerical Grade in CS2808: 65.67
Letter Grade in CS2808: D
Warning Note: Your grade is too low and needs improvements

Course number: CS3358
Exam 1: 72.30
Exam 2: 80.20
Exam 3: 99.10
Exam 4: 44.40
Exam 5: 75.20
Numerical Grade in CS3358: 72.74
Letter Grade in CS3358: C