Cairo University
Faculty of Engineering

1st year Communication
Computer

Student Report Management System

Number of students: 4

NAME	SEC	BN
1. خالد احمد سويلم	2	5
2. خالد جمال صابر	2	6
3.خالد خليفة عبدالحي	2	7
4. خالد عاطف سعد	2	8

➢ ***Introduction: -***

This project “Student Information management System” provides us a simple interface for maintenance of student information.it can be used by educational institutes or colleges to maintain the records of students easily. Achieving this objective is difficult using a manual system as the information is scattered, can be redundant, and collecting relevant information may be very time-consuming. All these problems are solved using this project.

Throughout the project, the focus has been on presenting information in an easy and Intelligible manner. The project is very useful for those who want to know about Student Information Management Systems and want to develop software/websites based on the same concept.

The project provides facilities like online registration and proves the creation of students thus, reducing paperwork and automating the record generation process in an educational institution.

➢ ***The proposed method:***

The program folder should at first contain a "student_report.txt" text file contains the data of the students each on a separate line, the data should be in the file in a specific order and comma-separated form. The order of the data in a single line is:

Rollno,name,dep,city,residential-area,zipcode,no_subjects,degrees,1,2…

And this is an example of how should be the data in the file look like:

2,Khaled Ahmed,eece,giza,Warak,550,1,95.7

By starting the program, the program creates a vector of objects of type student and after reading the students it modifies the report file to include for each student the grade and the percent of the total score.

Each student object contains the attributes for reserving: **name**, **rollno**, **grade**, **no. subjects**, **degrees** in subjects as a dynamic array, **department**; and an **address** structure contains: the **city**, **residential area** and **zip code**.

```
class student {
private:
    unsigned int rollno;
    unsigned int n;      //to store the number of subjects.
    float per;          //The total percent of the student.
    float * marks;     //A pointer of the dynamic array with n elements to store marks
    string grade;     //The total grade of the student.
    string dep;      //The department name of the student.
    string name;    //The name of the student.
    address a;     //The address of the student.
    void calculate();  //A function to calculate the total percent and the grade of the student.
    void creatmarks(){ marks = new float[n]; };  /*Creating the dynamic array in the heap to store the
                                                  degrees of the student after knowing the number of subjects*/
public:
    void st_output(ostream&);     //Outputs the data of the student to a stream
    void st_input(string s);     //Gets the data of a student from a string as a line, and the columns of the data are comm
a-separated.
    void st_input(bool f);            //Fills in a student object through the terminal.
    void show_data();          //Shows the record card of a single student onto the terminal.
    string rename();          //Returns the name of a student object.
    unsigned int rerollno(); //Returns the rollno of a student object.
};
```

1. ***st_output(ostream&)***

   This method outputs the record card of the student object to an output file stream for the student_report.txt file like the following example:

   1,Khaled Ahmed,EECE,Cairo,Hawamdia,1212,2,100,99,A,99.5

   And the only difference from the original record is that the grade and the percent are calculated and written to it.

```
void student::st_output(ostream& out){       //Outputs the student data into an output file stream in csv form
.
    out << rollno << ',' << name << ',' << dep << ','
        << a.city << ',' << a.res << ',' << a.zip << ',' << n << ',';
    for (unsigned int i = 0; i < n; i++)
        out << marks[i]<< ',';
    out << grade << ',' << per << '\n';
}
```

2. ***st_input(string s):***

   The line which contains the data of the student as illustrated above will be read as a string object and will be fed to that method which in turns will fill in the record card of that student by first creating a string stream and seed in its buffer with the string object line. Using a vector of strings named column, the line is divided into strings each referring to an attribute of the student object, after that the attributes got filled by the ordered string objects within the column vector.

```
void student::st_input(string line){  /*Fills in the data of a student object by a string line
                                      containing the data in a comma-separated form with a specific sequence as follows:
                                              rollno,st_name,dep.,city,res.area,zip_code,no_subjects,degree_in_sub1,,,,*/
    vector <string>(column);
    stringstream ss(line);       //Creating a string stream ss and initialize its buffer by string line.
    string temp;  //String variable to separate the data in the string stream.
    while (getline(ss, temp, ',')){
        column.push_back(temp);
    }
    rollno = stoi(column[0]);   /*Storing the rollno of the student form the first
                                column after changing it from a string into an integer value.*/

    name = column[1];           /*Storing the name of the student form the second column.*/
    dep = column[2];            /*Storing the department of the student form the second column*/
    a.city = column[3];         /*Storing the city of the student*/
    a.res = column[4];          //Storing the residential area of the student.
    a.zip = stoi(column[5]);   //Storing the zipcode of the student after converting it into an integer value.
    n = stoi(column[6]);       //Storing the number of subjects
    creatmarks();
    for (unsigned int i = 0; i < n; i++)
        marks[i] = stof(column[i + 7]);   //Storing the marks of the student.
    calculate();                        //Calculate the percent and optaining the grade of the student.
}
```

3. ***st_input(`bool f`):***

   This method fills in the student object trough terminal (console) interacting with the user to fill each attribute one by one, it takes every line input as a string and if nothing was entered the attribute remains for its original value if the record is modified not entered for the first time.

```
void student::st_input(bool f){  //The argument is 0 for modifying a student and 1 for adding a new student.
    string input;          //To store each input data from the user.
    cout << "\nEnter the rollno of the student\n\n>>";
    cin.ignore();
    if (getline(cin, input) && input != "")
        rollno = stoi(input);
    cout << "\nEnter the name of the student\n\n>>";
    if (getline(cin, input) && input != "")    //Getting the name of the student.
        name = input;
    cout << "Enter the department of the student:\n\n>>";    //Storing the department of the student.
    if (getline(cin, input) && input != "")
        dep = input;
    cout << "\nEnter the adress of the student\n\nCity: ";   //Storing the address of the student.
    if (getline(cin, input) && input != "")                  //city.
        a.city = input;
    cout << "\nResidential area: ";   //Storing the residential area of the student.
    if (getline(cin, input) && input != "")
        a.res = input;
    cout << "\nZip code: ";                         //Storing the zipcode of the student.
    if (getline(cin, input) && input != "")
        a.zip = stoi(input);
    cout << "\nEnter the student's degrees in the subjects \n";       //Storing the degrees in the subjects.
    if (f){   /*If it's adding a new student it will create a dynamic array for the degrees of the subject
               and if it is modifying an existing student nothing changes about the number of subjects.*/
        cout << "\n\nEnter the number of subjects to this student\n\n>>";
        if (getline(cin, input) && input != ""){
            n = stoi(input);
            creatmarks();  //Creating the dynamic array for storing the degree of each subject of the student
.
        }
    }
    cout << '\n';
    for (unsigned int i = 0; i < n; i++){        //Filling in the array of the degrees of the subjects.
        cout << "Subject #" << i + 1 << " : ";
        if (getline(cin, input) && input != "")
            marks[i] = stof(input);
    }
    calculate();  //Calculates the grade and the percent of the student.
}
```

4. *show_data():*

It outputs the data of the record card of a student object on the screen in the following form:

Rollno	Name	Dep.	City	Res	ZIP	No.subjects	Grade	Percent	Degrees
2	Khaled Ahmed	EECE	Cairo	Hawamdia	550	5	A	92.8%	50 95 96 85 98

```
void student::st_output(ostream& out){      //Outputs the student data into an output file stream in csv form
.

    out << rollno << ',' << name << ',' << dep << ','

        << a.city << ',' << a.res << ',' << a.zip << ',' << n << ',';

    for (unsigned int i = 0; i < n; i++)

        out << marks[i]<< ',';

    out << grade << ',' << per << '\n';

}
```

5. *The rename() and rerollno() methods:*

These methods return the name and rollno of a student object to use them in some user-defined functions.

```
string student::rename(){ return name; }      //Returns the name of a student object.
unsigned int student::rerollno(){ return rollno; }  //Returns the rollno of a student object.
```

6. *Calculate():*

This method calculates the grade and the percent of the total score of the student object.

```
void student::calculate(){      /*Creating the dynamic array in the heap to store the
                                                  degrees of the student after knowing the number of subjects
*/
    per = 0;
    for (unsigned int i = 0; i < n; i++)  /*Summing all the degrees of the student.
                                            assuming that for each subject the full score is 100.*/
        per += marks[i];
    per /= n;        //Calculating the total percent.
    if (per >= 85)       //getting the grade of the student.
        grade = 'A';
    else if (per >= 75)
        grade = 'B';
    else if (per >= 65)
        grade = 'C';
    else if (per >= 50)
        grade = 'D';
    else
        grade = 'F';
}
```

Note: One thing about the reading process that the records must be in the form of a comma-separated data and the specified order and any change would result in a program crash.

And there is some user-defined function we used to help the program in the required functionalities.

1. *Lower*():
   Turns a string object to lower case.

```
void lower(string& stt){ for_each(stt.begin(), stt.end(), [](char & c){c = tolower(c);});}
```

2. *int display_required(string&):*
   Returns the index of a student object in starr by it's name.

```
int display_required(string& rat){
    bool found = 0;
    string temp;
    for (int i = 0; i < starr.size(); i++){  //Looping over starr.
            temp = starr[i].rename();
            lower(temp);
        if (temp == rat){
            cout << "\nfound!\n\n";
            found = 1;
            return i;
        }
    }
    if (!found){ //If not found.
        cout << "\nStudent not found !\n\n";
        return -1;
    }
}
```

3. *int display_required(int):*
   Returns the index of a student object in starr by it's rollname.

```
int display_required(int rat){
    bool found = 0;
    for (int i = 0; i < starr.size(); i++){//Looping over starr.
        if (starr[i].rerollno() == rat){
            cout << "\nfound!\n\n";
            found = 1;
            return i;
        }
    }
    if (!found){//If not found.
        cout << "\nStudent not found !\n\n";
        return -1;
    }
}
```

Now after reading the file and filling all student objects of the vector **starr** a menu shows up to the screen with the operations the user can choose from them and it contains the following:

1. Display student data:

In this option the program outputs the data of a student, looping inside the vector of the student objects and for each student prints, it's data using the method .show_data().

2. Add a student:

It adds a student data to the report file and appends the vector containing the student objects with a new object of the new student, after appending the vector, it uses the method st_input() for the last student in the vector to be filled, after filling its attributes the program appends the report file with the new student.

3. Modify a record

Entering the rollno of the student to be modified, the loop scans for all objects in the vector having the entered rollno, if found; the program shows its data and starts to ask the user to enter the attributes if the user doesn't want to change any attribute, he could just press enter to skip it, after that the program modifies the report file with new data and the object stored in the victor got modified.

4. Search for a student by name

It takes the name of the student as an input, then passes it to the formal parameter of the user-defined function display_required(string name), the function loops over the vector of student objects, if the student exists, the function returns the index of that object in the vector, if not; the function returns -1 to activate if statement; and prints "not found!".

Note: for the search to be not case sensitive, we introduced the function string lower(string&); which turns a string object into lower case.

5. Search for a student by num

It takes the rollno of the student as an input, then passes it to the formal parameter of the user-defined function display_required(int rollno), the function loops over the vector of student objects, if the student exists, the function returns the index of that object in the vector, if not; the function returns -1 to activate if statement; and prints "not found!".

6. Calculate student grade

Shows the grade and the total score of a specific student, the student is searched by its rollno or by its name, it's up to the user to choose.

7. Delete a record

It deletes a student record from the file and the vector where the objects of the students' records are stored, taking care of the order of the objects and records in the file to be unchanged.

After deleting the object from the vector, it deletes it from the file.

➢ ***Conclusion:***

According to each function, the program should do:

1. For reading the student information from the file: it works well in reading the student data from the file but sensitive for any extra space between the entries, so the user is so restricted to the style of comma-separated entries.

   One way to solve that problem is to remove any spaces between the commas separating the entries taking much care that some spaces could be found in the name of the student or the city or the residential area, that we don't want them to be removed.

2. Displaying student information: The arrangement of the columns show some difficulty for the user to get the required data easily, and since the students have a different number of subjects, the resolutions of the console screen even if they were set to wrap the content the output could exceed the screen, so the program should take some care of that.

   A suggestion for a problem like would be to track the length of the output for each student and if it exceeds the screen resolutions or the console window dimensions, it makes a new line with the rest of the data, showing above them the headers of the table.

3. Searching for a student by name or by rollno: The searching functionality works so well in searching with the rollno, and for searching for the name the functionality is not case sensitive.
   Also searching by rollno works well
   One such thing to notice that if the name or the rollno of the student is repeated in the file, or if two students have similar names the program recognizes the first to exist in the file.

   To solve such a problem we could easily make the program go through the entire file searching for that name or rollno and creates an array with the indexes of the students with that name.

4. Modify a student record: the functionality works as wanted; the user decides which data to be changed from the record, and after the user finishes the program modifies the file for the new student list.

5. Calculate the grade of a student: it also works well, but in calculating the final score, the program assumes that the total degree for each student is 100, and this not the case in the real word.

6. Delete a student: it also works well, remove the student from the storing vector and from the file, taking care of the order of the students in the file.

   A solution to this problem is to change the marks matrix into a two-dimensional array

   Marks[n][2], to store for each subject the degree and the total score of that particular subject.

Some more interesting modifications could be add to the system, one of the most wonderful among them is to implement the system as a "Graphical user interface", which will be more user friendly and offers a very small error opportunity.