ADNAN MENDERES UNIVERSITY
CSE 102 Algorithms and Programming

Term Project
(Student Grade Management System)

151805005 Alper ÇALIŞKAN
161805025 Kemal YILDIRIM
161805059 Oğuz BARDAKÇI

Contents

Description of the problem:

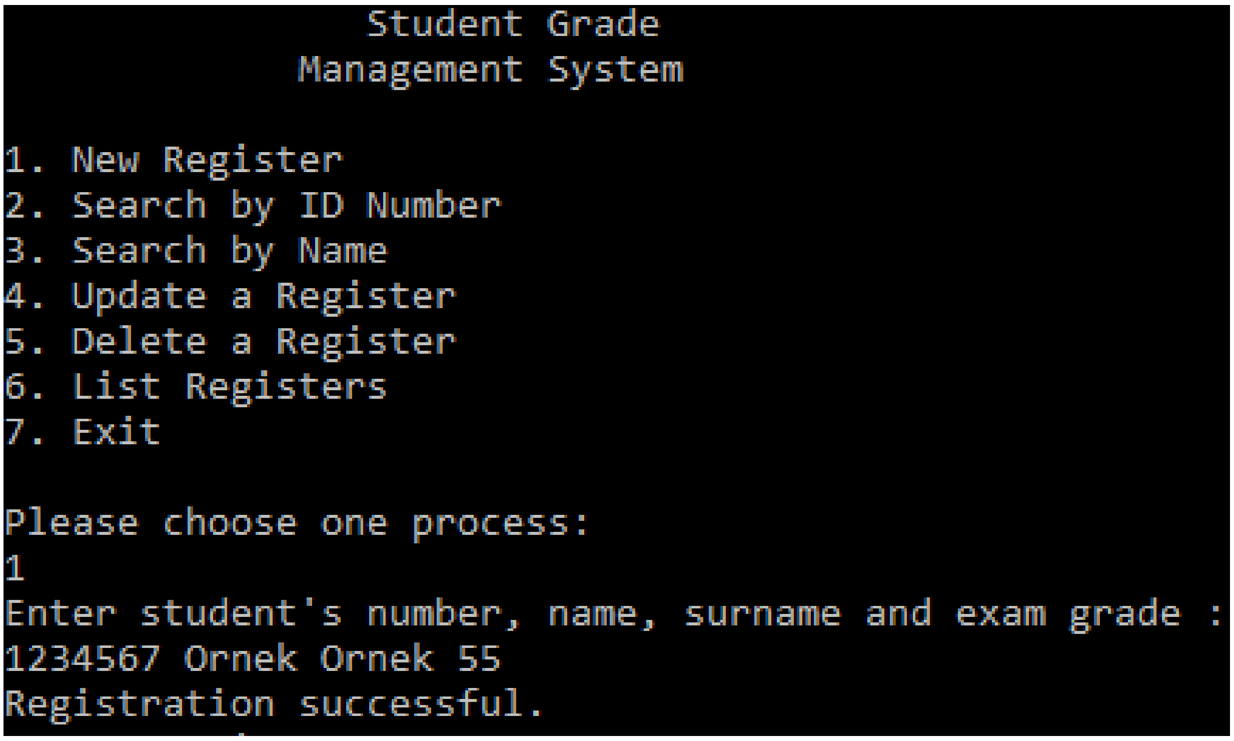
We need a program that hold the students student number, student name, student surname and exam grade and after then saves that records in a text file. Also these records can be updated, deleted, listed and viewed by student number and student name.

Major steps for solving the problem:

The biggest step was getting together and being a team. Beside that in the program we thought and searched a lot about updating and deleting a register. We can say that was our biggest difficulty.

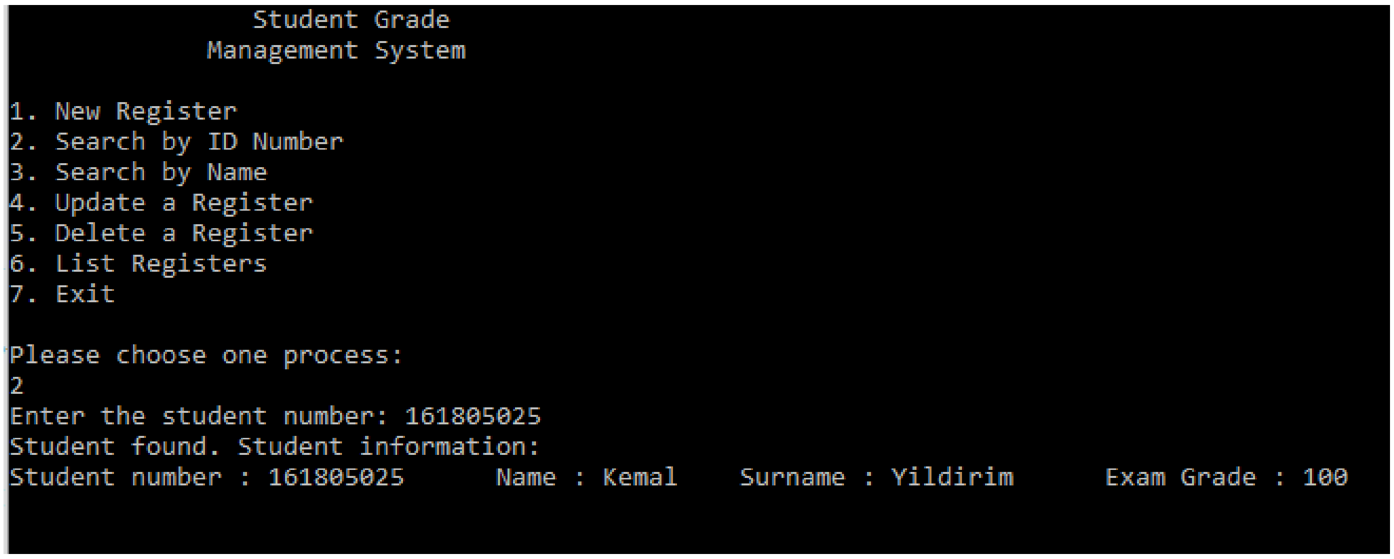
Input/output of the program:

1-New Register:



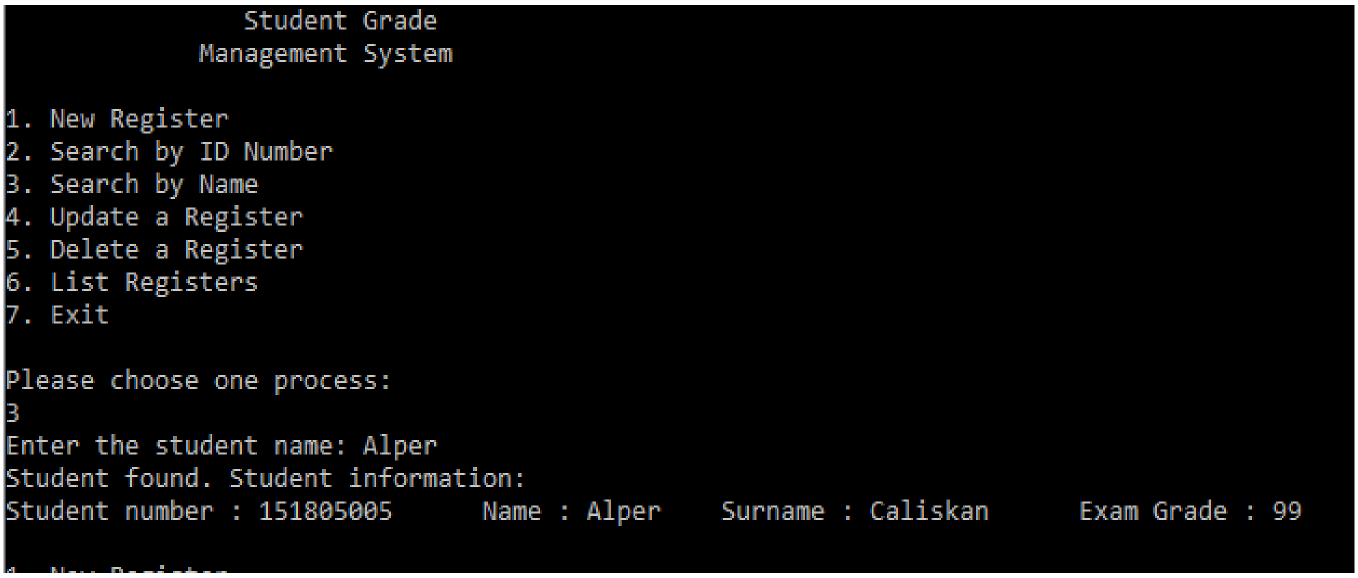
```
                    Student Grade
                Management System

1. New Register
2. Search by ID Number
3. Search by Name
4. Update a Register
5. Delete a Register
6. List Registers
7. Exit

Please choose one process:
1
Enter student's number, name, surname and exam grade :
1234567 Ornek Ornek 55
Registration successful.
```

2-Search by Number:



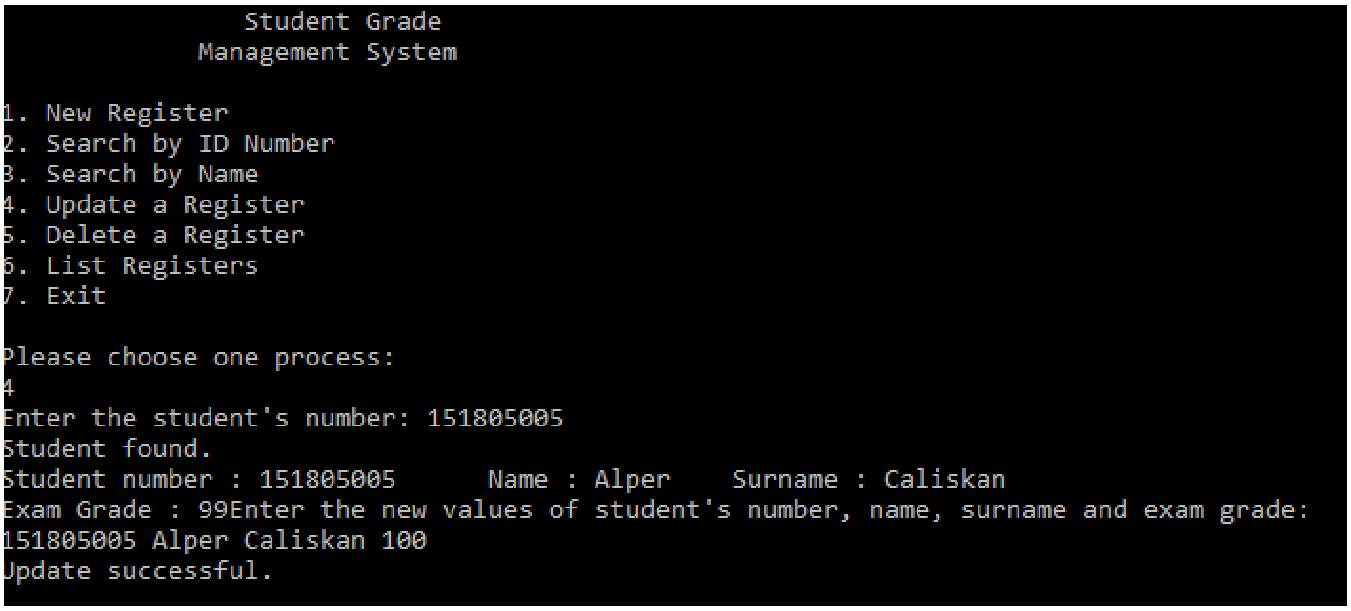
```
                Student Grade
             Management System

1. New Register
2. Search by ID Number
3. Search by Name
4. Update a Register
5. Delete a Register
6. List Registers
7. Exit

Please choose one process:
2
Enter the student number: 161805025
Student found. Student information:
Student number : 161805025      Name : Kemal    Surname : Yildirim      Exam Grade : 100
```

3-Search by Name:



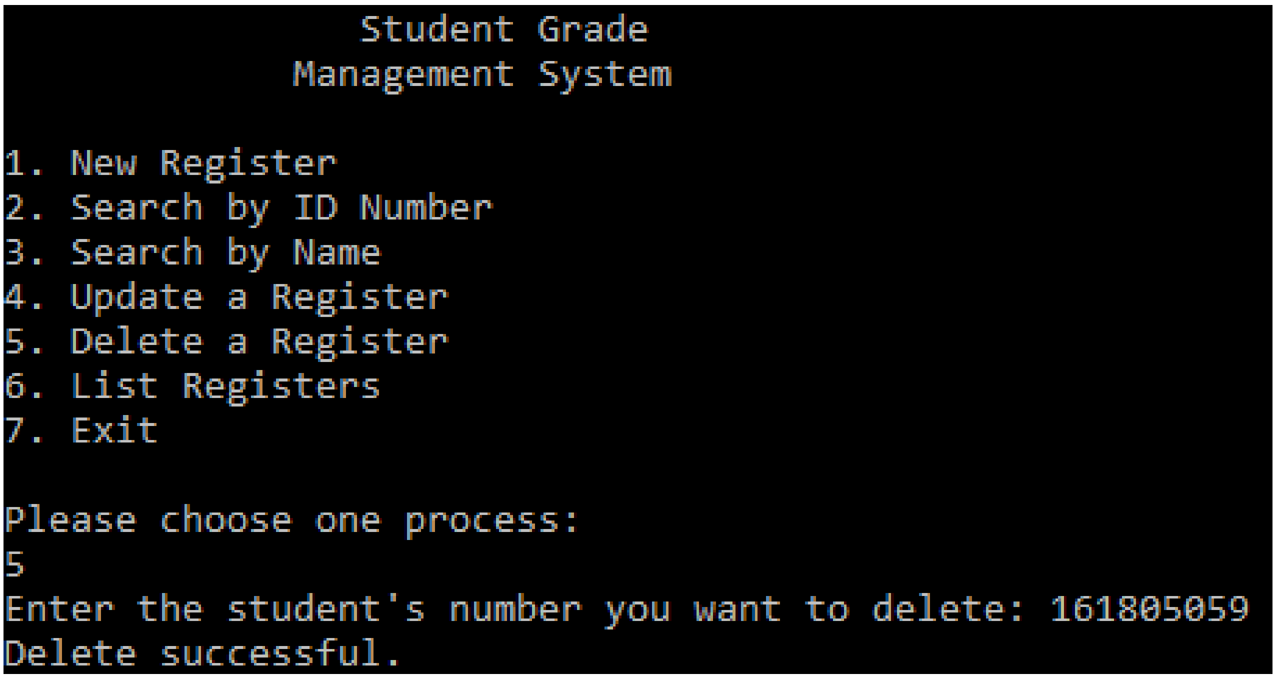
```
                Student Grade
             Management System

1. New Register
2. Search by ID Number
3. Search by Name
4. Update a Register
5. Delete a Register
6. List Registers
7. Exit

Please choose one process:
3
Enter the student name: Alper
Student found. Student information:
Student number : 151805005      Name : Alper    Surname : Caliskan      Exam Grade : 99
```

4-Update Register



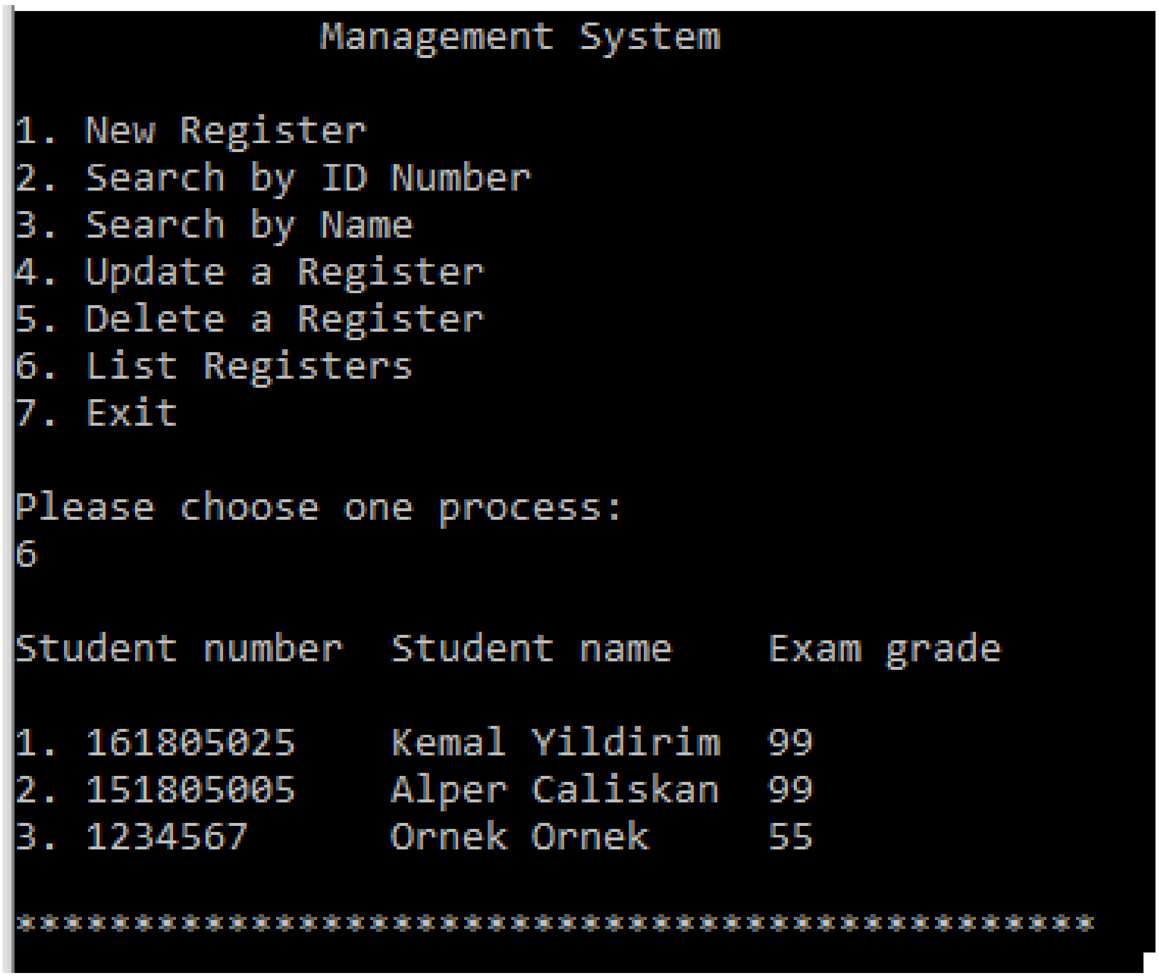
```
                Student Grade
             Management System

1. New Register
2. Search by ID Number
3. Search by Name
4. Update a Register
5. Delete a Register
6. List Registers
7. Exit

Please choose one process:
4
Enter the student's number: 151805005
Student found.
Student number : 151805005      Name : Alper    Surname : Caliskan
Exam Grade : 99Enter the new values of student's number, name, surname and exam grade:
151805005 Alper Caliskan 100
Update successful.
```

5-Delete Register



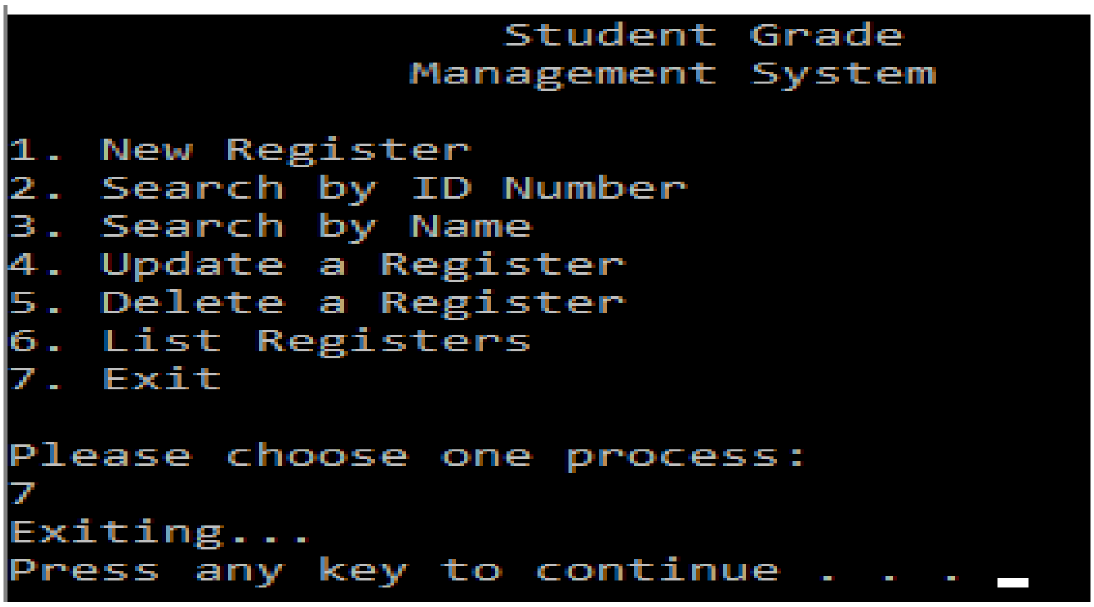
```
                 Student Grade
              Management System

1. New Register
2. Search by ID Number
3. Search by Name
4. Update a Register
5. Delete a Register
6. List Registers
7. Exit

Please choose one process:
5
Enter the student's number you want to delete: 161805059
Delete successful.
```

6-List Registers



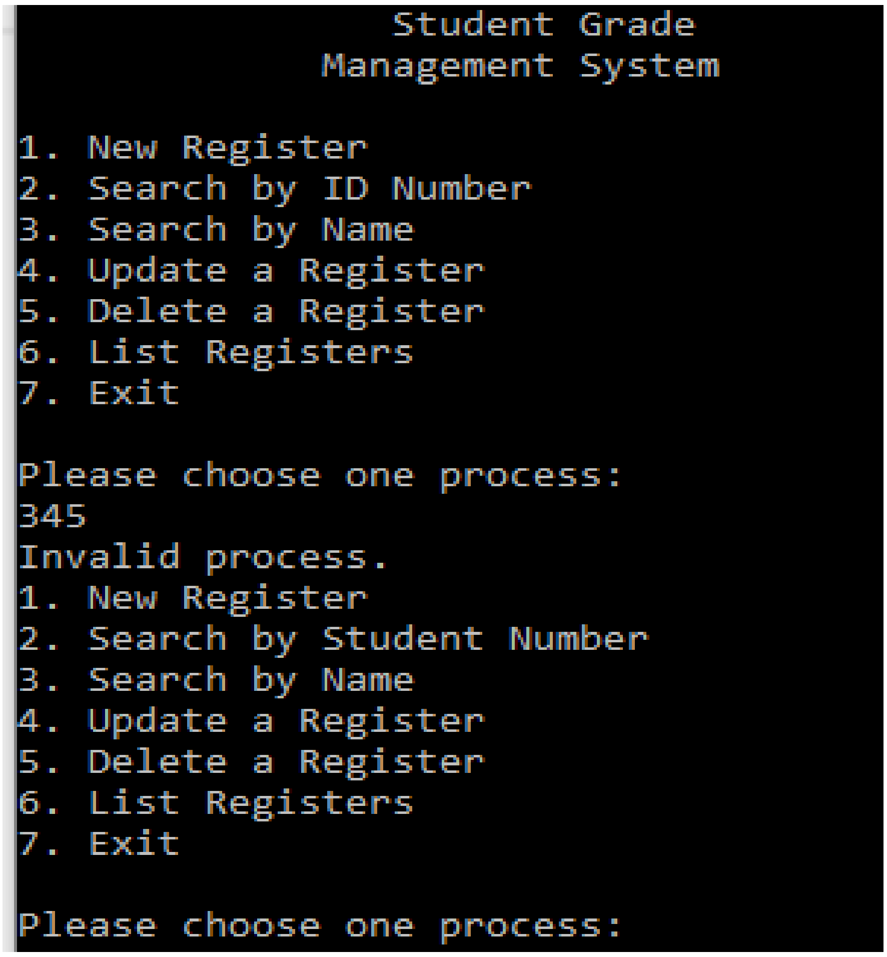
```
             Management System

1. New Register
2. Search by ID Number
3. Search by Name
4. Update a Register
5. Delete a Register
6. List Registers
7. Exit

Please choose one process:
6

Student number  Student name    Exam grade

1. 161805025    Kemal Yildirim  99
2. 151805005    Alper Caliskan  99
3. 1234567      Ornek Ornek     55

*********************************************
```

7-Exit



```
                    Student Grade
                Management System

1. New Register
2. Search by ID Number
3. Search by Name
4. Update a Register
5. Delete a Register
6. List Registers
7. Exit

Please choose one process:
7
Exiting...
Press any key to continue . . .
```

8-Invalid processes



```
                    Student Grade
                Management System

1. New Register
2. Search by ID Number
3. Search by Name
4. Update a Register
5. Delete a Register
6. List Registers
7. Exit

Please choose one process:
345
Invalid process.
1. New Register
2. Search by Student Number
3. Search by Name
4. Update a Register
5. Delete a Register
6. List Registers
7. Exit

Please choose one process:
```

Description of how we test this program:

We simply debugged the program a lot. We checked almost every input and output and we asked for help to our friends in the class.

Contribution of each team member (who did what):

After the project given us, our team leader arranged a meeting to two days later. At the meeting, by our team leader the outline has been prepared. After that, we shared that outline and everyone did their part as possible. We arranged another meeting and at that meeting we talked about the parts which we cannot do and finished the source code. The next three days, we write the project report.

Source code:

```
/*
Adnan Menderes University Faculty of Engineering
Computer Engineering Department
Algorithm and Programming Term Project

Group Members :

161805025 Kemal Yildirim
151805005 Alper Caliskan
161805059 Oguz Bardakci

*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#pragma warning(disable:4996)

struct studentData            //global struct defination
{
     int number; char name[21]; char surname[21]; int
examgrade;

}student; //struct variable



void newRegister()
{
     FILE *fPtr;


     if ((fPtr = fopen("students.txt", "ab+")) == NULL) //ab+
takes new register to end of the file
     {
          puts("File cannot be opened.");
     }
     else
     {
          puts("Enter student's number, name, surname and exam
grade :");
          scanf("%d%20s%20s%d", &student.number, student.name,
student.surname, &student.examgrade); // it wants the student
number twice.
```

```
          fwrite(&student, sizeof(struct studentData), 1,
fPtr);                        // we couldnt find a way out.
          puts("Registration successful.");

     }
     fclose(fPtr);
}

void searchByNumber()
{
     int request;
     printf("Enter the student number: ");
     scanf("%d", &request);
     FILE *fPtr;
     if ((fPtr = fopen("students.txt", "rb+")) == NULL)
     {
          puts("File cannot be opened.");
     }
     else
     {
          while (fread(&student, sizeof(struct studentData), 1,
fPtr))
          {
               if (student.number == request)
               {
                    printf("Student found. Student
information:\nStudent
number : %d\tName : %s\tSurname : %s\tExam Grade : %d\n\n",
student.number, student.name, student.surname,
student.examgrade);
                    break;
               }

          }
          fclose(fPtr);
          puts("");
     }
     }

void searchByName()
{
     char request[21];
     printf("Enter the student name: ");
     scanf("%s", request);
     FILE *fPtr;
     fPtr = fopen("students.txt", "rb+");

     while (fread(&student, sizeof(struct studentData), 1,
fPtr))
     {
          if (strcmp(student.name, request) == 0)
          {
               printf("Student found. Student
information:\nStudent
number : %d\tName : %s\tSurname : %s\tExam Grade : %d\n\n",
student.number, student.name, student.surname,
student.examgrade);
               break;
          }
     }
     fclose(fPtr);
}

void updateRegister()
{
     FILE *fPtr;
     if ((fPtr = fopen("students.txt", "rb+")) == NULL)
     {
          puts("File cannot be opened.");
     }
     else
     {
          int intrequest;
          int record = 0; //file begins with 0th line and
continues like 1-2-3 so ..
          printf("Enter the student's number: ");
          scanf("%d", &intrequest);
          while (fread(&student, sizeof(struct studentData), 1,
fPtr))
          {
               if (student.number == intrequest)
               {
                    puts("Student found.");
                    printf("Student
number : %d\tName : %s\tSurname : %s\nExam Grade : %d",
student.number, student.name, student.surname,
student.examgrade);
                    puts("Enter the new values of student's
number, name, surname and exam grade: ");
                    scanf("%d%s%s%d", &student.number,
student.name, student.surname, &student.examgrade);
                    fseek(fPtr, sizeof(struct studentData) *
record, SEEK_SET); //takes the pointer to beginning of the
line
                    fwrite(&student, sizeof(struct
studentData), 1, fPtr);
                    puts("Update successful.\n");
                    break;
               }
               record++; // if dont find the student increases
the line count
```

```
          }
          fclose(fPtr);


     }
}

void deleteRegister()
{
     FILE *fPtr, *temp;
     if ((fPtr = fopen("students.txt", "rb")) == NULL)
     {
          puts("File cannot be opened.");
     }
     else
     {
          int request;
          temp = fopen("Temp.txt", "wb+");
          printf("Enter the student's number you want to 
delete: ");
          scanf("%d", &request);
          while (fread(&student, sizeof(struct studentData), 1, 
fPtr))
          {
               if (student.number != request)
               {
                    fwrite(&student, sizeof(struct 
studentData), 1, temp);
               }
          }
          fclose(fPtr);
          fclose(temp);
          remove("students.txt");
          rename("Temp.txt", "students.txt");
          puts("Delete successful.");
     }
}

void listRegisters()
{
     FILE *fPtr;
     if ((fPtr = fopen("students.txt", "rb+")) == NULL)
     {
          puts("File cannot be opened");
     }
     else
     {
          puts("\nStudent number\tStudent name\tExam grade\n");
          int i = 1;
          while (fread(&student, sizeof(struct studentData), 1, 
fPtr))
```

```
          {
               printf("%d. %d\t%s %s\t%d\t\n", i,
student.number, student.name, student.surname,
student.examgrade);
               i++;
          }

     printf("\n*********************************************\n
\n");
          fclose(fPtr);
     }

}

int main(void) {

     unsigned int process;

     puts("\t\tStudent Grade\n\t     Management System\n");
     puts("1. New Register\n2. Search by ID Number\n3. Search
by Name\n4. Update a Register\n5. Delete a Register\n6. List
Registers\n7. Exit");
     puts("\nPlease choose one process: ");
     scanf_s("%u", &process);

     while (1)
          {
          switch (process)
          {
          case 1:
               newRegister();
               break;

          case 2:
               searchByNumber();
               break;
          case 3:
               searchByName();
               break;
          case 4:
               updateRegister();
               break;
          case 5:
               deleteRegister();
               break;
          case 6:
               listRegisters();
               break;

          case 7:
               puts("Exiting...");
```

```
                exit(EXIT_SUCCESS);
                break;
            default:
                puts("Invalid process.");
                break;
            }
            puts("1. New Register\n2. Search by Student 
Number\n3. Search by Name\n4. Update a Register\n5. Delete a 
Register\n6. List Registers\n7. Exit");
            printf("\nPlease choose one process: ");
            scanf_s("%d", &process);
            }
        }
```