Cairo University
Faculty of Computers and Artificial Intelligence
Computer Science Department

Programming-1 CS112 **Assignment 4** Dr. Amin Allam

1 Students database

⇒ Create a student structure as follows:

```
const int MAX_NAME=14; // Assume a name contains at most 14 characters
enum Gender{MALE,FEMALE};
struct Date{int year; int month; int day;};

struct Student
{
   int    id;
   char   first_name[MAX_NAME+1];
   char   last_name[MAX_NAME+1];
   Date   birth_date;
   Gender gender;
   double gpa;
};
```

⇒ Define a function `InputStudent()` which takes a student data from the user and saves it into a `Student` object.
⇒ Define a function `OutputStudent()` which prints a student data in one line.

```
void InputStudent(Student* s);   // s is a pointer to one object
void OutputStudent(Student* s);
```

Each student data should be output in *one line* in the following order:
`id first_name last_name birth_date gender gpa`
The following is an example of student data input and output:

```
20090111 Aly Ahmed 26/5/1992 Male 3.4
```

⇒ Define a function `InputAllStudents()` which inputs `n` students from the user and saves them in a dynamic array which was previously allocated in `main()`.
⇒ Define a function `OutputAllStudents()` which prints all student data, each student in one line.

```
void InputAllStudent(Student* s, int n);    // s is a pointer to dynamic
void OutputAllStudent(Student* s, int n);   //  array of n objects
```

⇒ Define a function `BirthLess()` that takes 2 student pointers and returns `true` if the first student is born before the second student.
⇒ Define a function `GpaGreater()` that takes 2 student pointers and returns `true` if the gpa of the first student is greater than the gpa of the second student.

```
1 bool BirthLess(Student* a, Student* b);
2 bool GpaGreater(Student* a, Student* b);
```

⇒ Define a function `SortStudentsByBirthDate()` that sorts `n` students by increasing birth date (the student with the earliest birth date should be first). The function should call `BirthLess()`.
⇒ Define a function `SortStudentsByGpa()` that sorts `n` students by decreasing gpa (the student with the highest gpa should be first). The function should call `GpaGreater()`.

```
1 void SortStudentsByBirthDate(Student* s, int n);
2 void SortStudentsByGpa(Student* s, int n);
```

⇒ Define a function `SearchStudentId()` that takes `n` students and a student id as parameters and returns a pointer to a student having this id or `0` if not found.
⇒ Define a function `SearchStudentFirstName()` that takes `n` students and a C-string as parameters and returns a pointer to any student having this first name or `0` if not found.

```
1 Student* SearchStudentId(Student* s, int n, int id);
2 Student* SearchStudentFirstName(Student* s, int n, char* name);
```

⇒ The program should start by taking the number of students `n` from the user. The program should create a dynamic array of `n` students.
⇒ Then, the program should take the full data of each of the `n` students from the user.
⇒ Then, the program should output a list of choices for the user as follows:

```
1) Output all students data.
2) Sort students by increasing birth date.
3) Sort students by decreasing gpa.
4) Search students by id.
5) Search students by first name.
6) Exit the program.
```

⇒ If the user inputs `1`, the program should output the full data of the `n` students, each student in one line, then reprints the list of choices.
⇒ If the user inputs `2`, the program should sort the `n` students by increasing birth date then outputs all sorted students, then reprints the list of choices.
⇒ If the user inputs `3`, the program should sort the `n` students by decreasing gpa then outputs all sorted students, then reprints the list of choices.
⇒ If the user inputs `4`, the program should take integer from the user, searches for a student having this id, prints the student full data if it exists, then reprints the list of choices.
⇒ If the user inputs `5`, the program should take a C-string from the user, searches for a student having this first name, prints the student full data if it exists, then reprints the list of choices.
⇒ If the user inputs `6`, the program should release the dynamic array and exit.