Project Ideas

You can select one of these two projects.

Project 1: Telephone Book

It is required to write a program that will create and maintain a phone directory. Each entry of the directory will include the **first name, last name, address, phone number and email**. The program **MUST use nested structure** (Hint: address inside contact: **street, flat, city**). Your program should allow the user to execute the following commands:

1- **LOAD:** (Read from file) This command reads in a file name and loads the phone directory from this file to an array of contacts. The file is a text file that is comma delimited in which each entry is found on a separate line. Each entry must contain the first name, last name, street, flat, city and phone number, email. An example of one line in the file is as follows:
Ahmed,Mahmoud,26 Elhoreya Street,15,Alexandria,4876321, ahmed@gmail.com

2- **QUERY:** (Search) The system should process a request by the user to look up information about a specific entry. The user must supply the person's first name, and the system should provide the first name, last name, address and phone number for all users of that first name.

3- **ADD:** The system should prompt the user field by field for the data of a single record and add it to the array of contacts.

4- **MODIFY:** The user should be prompted for the first name and last name (assuming first and last name is a unique combination). Then the user should be prompted field by field to modify the information for one of the records. If the user left a field empty, that means that the user doesn't want to update this field.

5- **DELETE:** The user should be prompted for the first name and last name (assuming first and last name is a unique combination) and a record associated with that name should be deleted from the array of contacts.

6- **PRINT:** (Sort) Print the entire array of contacts on the screen, in sorted order. Entries should be sorted according to first name then last name.

7- **SAVE:** (Write to file) Save the array of contacts by writing it out to an external file in a format similar to that explained in the Load command (the same file). Note that: Functions from 2~6 are operating on the array of contacts.

8- **QUIT:** Exit (without saving the directory in the external file). You should display a warning to the user about all of his/her changes will be discarded.

Notes

- The program should check for errors and print appropriate messages (e.g. trying to delete an entry that is not in the file, or trying to search for a contact that doesn't exist).
- Every task of the above tasks should be a separate function.
- You shall use global variable, it will help you a lot.
- You should do a menu to enable an easy access for all of these functions.

Bonus

- In task 4, 5: You should search for a contact by first name only, and then you should print a list to the user to choose one of the results to edit/delete.
- The program should validate all of entered data.
    - Validate a number in the phone number and in the flat.
    - Validate a sting in the first name, last name, street, city
    - Validate the email address (example@domain.com).

Project 2: Students Management

It is required to write a program that will create and maintain a Student Scores and other management tools. Each entry of the directory will include the **name, id, contact information, age and score**. The program **MUST use nested structure** (Hint: contact information inside student: **city, mobile number, email**). Your program should allow the user to execute the following commands:

1- **LOAD:** (Read from file) This command reads in a file name and loads the student directory from this file to an array of students. The file is a text file that is comma delimited in which each entry is found on a separate line. Each entry must contain the name, id, city, mobile, email, age and score. An example of one line in the file is as follows:
Ahmed Mahmoud,2626,Alexandria,01003658472, ahmed@gmail.com,19,96.35

2- **QUERY:** (Search) The system should process a request by the user to look up information about a specific entry. The user must supply the student id, and the system should provide all the information.

3- **ADD:** The system should prompt the user field by field for the data of a single record and add it to the array of students.

4- **DELETE:** The user should be prompted for the id and a record associated with that name should be deleted from the array of students.

5- **PRINT:** (Sort) Print the entire array of students on the screen, in sorted order. Entries should be sorted according to the id. (id is unique)

6- **STATISTICS:** (Max, Min and Average) Print the names and the ids of the top 5 students in the class. Then print a statistics about maximum score, minimum score and the average score.

7- **UPDATE_SCORE:** The user should be prompted for the id. Then the user should be prompted to enter the new score for this student.

8- **SAVE:** (Write to file) Save the array of students by writing it out to an external file in a format similar to that explained in the Load command (the same file). Note that: Functions from 2~8 are operating on the array of contacts.

9- **QUIT:** Exit (without saving the directory in the external file). You should display a warning to the user about all of his/her changes will be discarded.

Notes

- The program should check for errors and print appropriate messages (e.g. trying to delete an entry that is not in the file, or trying to search for a contact that doesn't exist).
- Every task of the above tasks should be a separate function.
- You shall use global variable, it will help you a lot.
- You should do a menu to enable an easy access for all of these functions.

Bonus

- Implement a **HALF_LOAD** function: The user can select this function that would add 0.2 to all GPA's less than 2.
- Implement **NORMALIZE** function: The user can select this function that would calculate the average and then add a 5% of it to all of the students. Take care that maximum score is 100.
- The program should validate all of entered data.
    - Validate a number in the mobile number, age and the id.
    - Validate a sting in the name and city.
    - Validate the email address (example@domain.com).
    - Validating a real number in the score is a plus (extra).

General

Polices

- **Late submissions will not be accepted**.
- You should Work in **groups of two**.
- No time will be allowed at the lab during the discussion for any modifications.
- Cheating Policy: All actual programming should be an independent effort. If any kind of cheating is discovered, penalties will apply to the participating students by zero in the project, so delivering a nonworking program is so much better than delivering a copy.

Deliverables

- Report that will contain description for your work, sample runs for the program, user manual (how to use the system), and main algorithms used (search algorithm, sort algorithm).
- Deadline will be in the week starting from **24/12/2016** and Discussion date will be **determined later**.

Good luck isA ☺