	CS214 - Data Structures and Algorithms **Assignment No. 2** **(CLO 3)** **Dr. Mohammad Imran**	**Total Marks: 25** **Submission deadline:** **8:00 PM on 23rd October**

Instructions:

- **Submit the whole project, as one zipped file; the filename should be in the following format: REGID_Assign2.zip**
- **Code should be properly indented and well commented.**
- **The program must perform error checking and boundary condition testing.**
- **Your submissions will be graded on the basis of functionality (20 marks) and use of best practices in coding (5 marks).**
- **Assignments submitted late will NOT be marked.**
- **Detection of copying/cheating between students will result in award of zero marks to both students.**

Objectives:

The objectives of this assignment are to help you in:

(i) sharpening your skills in programming, especially in use of pointers and dynamic memory management.

(ii) getting a deeper understanding of circular and doubly linked lists.

An online freelancing platform, called **DevProj,** needs to record data of developers and their projects. You have to develop a solution for the platform making use of efficient data structures. The program should allow the user to traverse through the list of developers forward and backwards. Furthermore, the program should enable the user to see the projects for the current developer. The user does **not** need to traverse through the projects for a given developer in both directions, but the program should be able to again show the first project of the same developer after showing the last project.

The following data needs to be stored for each developer:

- Developer ID
- Name
- Specialization (Web, Mobile, Desktop)
- Number of projects

The data that the program should store about a project includes:

- Project ID
- Duration
- Cost
- Status (completed, in progress, cancelled)

The interface of the program should allow the user to choose a command from a menu as explained below:

```
Please choose a command:
d    Add a developer
p    Add a project
s    Search for a developer
t    Search for a project
v    View developers
f    Show next developer
r    Show previous developer
n    Show next project
e    exit
```

When the user presses '**d**', the program should ask the user to input the data required for the developer, and add the developer at the start of the developer list. The main menu is then shown again.

When the user presses '**p**', the program should ask the developer ID for whom the project will be added. Then the program should search for the developer. If found, then the program should print his/her details and ask the user to enter details of the new project. The new project should be added to the start of the list. If the developer is not found, the program should show an error message. In both cases, the user is again taken to the main menu.

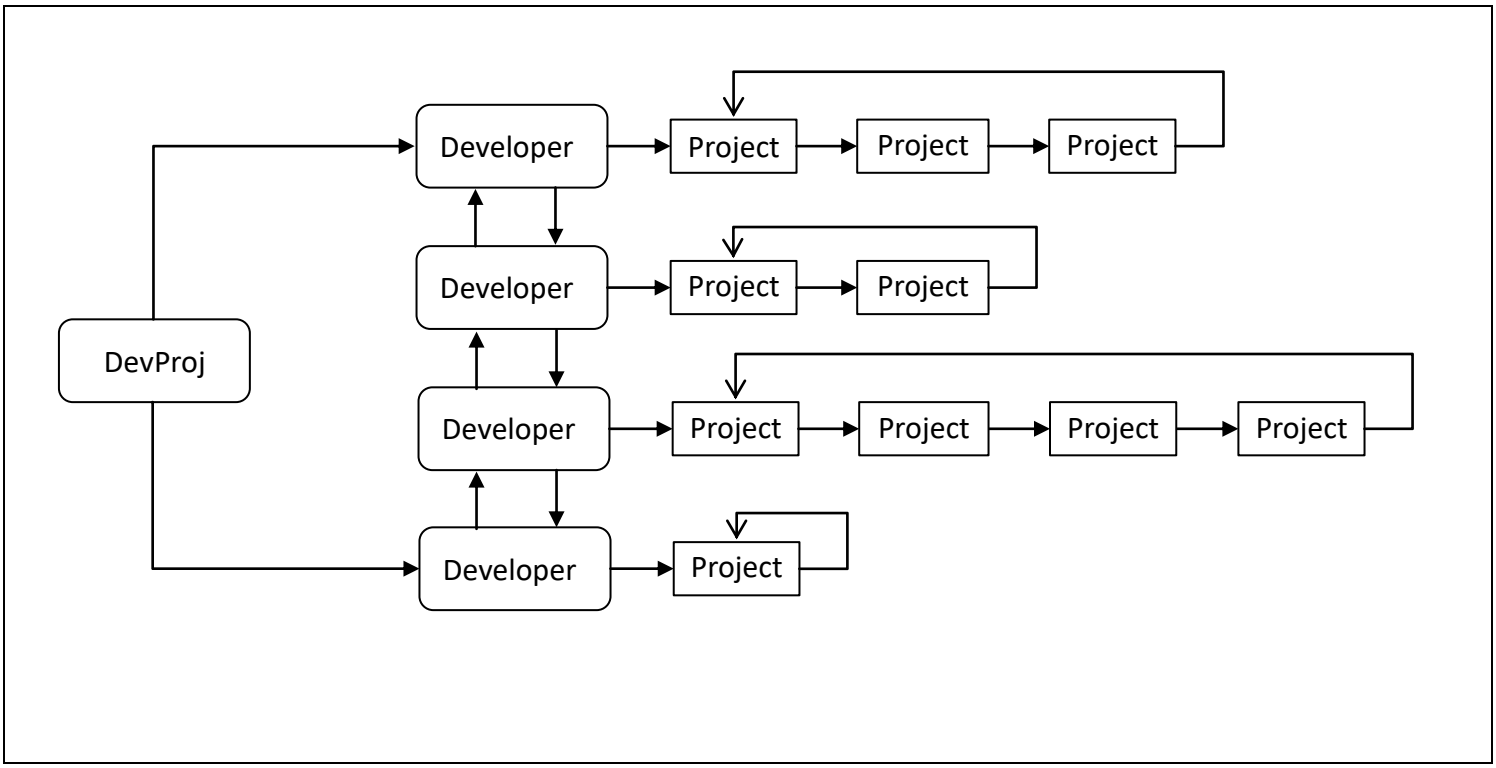
When the user presses '**s**' the program should ask the user to enter the ID of the developer. If the developer is found, his/her details should be shown. The details of the first project of this developer are also shown. The user should be able to see the next project of the same developer by pressing '**n**'. If the user is on last project for the current developer then pressing '**n**' should take him/her again to the first project of the current developer.

When the user presses '**t**', the program should ask the user to enter the project ID to search. The program should then search projects of all developers one by one. If the project is found, then the details of the developer and the project are shown. Otherwise, an error message is displayed and the user is taken to main menu again.

When the user presses '**v**', the first developer's data is shown. If the user presses '**f**', the next developer is shown. If the user is on last developer and '**f**' is pressed then there should be an error message saying that this is the last developer. If the user presses '**r**', the previous developer's data is shown. If the user is seeing the first developer and '**r**' is pressed then an error message should be shown that this is the first developer.

Looks difficult? It is not! Let me give you a starting point. First you need to create a doubly linked list of developers, and then for each developer you'll need a circular linked list of projects as shown in the figure. At the highest level, you'll need a class for DevProj which has the head and tail pointers for the developers linked list. To help you further, here are the classes that you'll need to make:

```
class DevProj {     // The doubly linked list of developers
    Developer *head;
    Developer *tail;

    // member functions


    // Function for adding a developer to the doubly linked list
    void AddDeveloper (Developer *dev)
    {
    }

    // Function for adding a project to a developer
    void AddProject (Developer *dev, Project *proj)
    {
    }

};

class Developer {   // The node for doubly linked list of developers
    int Dev_ID;
    char Name[30];
    char Speciality[20];
    int  Project_Count;

    Developer *next;// This is a node of doubly linked list, so we need both
    Developer *prev;// next and previous pointers

    Project *head;  // Each developer node contains head and tail pointers
    Project *tail;  // for the circular linked list of its projects

    friend class DevProj;

    // member functions

    // Function for adding a project to the circular linked list
    Void AddProject(Project *proj)
    {
    }


    // Function for printing the data of this developer
    void Print ()
    {
    }

};

class Project {     // The node for circular linked list of projects
    int Project_ID;
    int Duration;
    int Cost;
    char Status[10];

    Order *next;
    friend class Developer;

    // member functions


    // Function for printing the data of this project
    void Print ()
    {
    }

};
```


DevProj
Developer
Developer
Developer
Developer
Project
Project
Project
Project
Project
Project
Project
Project
Project
Project