Assignment-II

Employee Management System

In this assignment you will create an "Employee Management System" for all the employees of "XYZ Corporation". You must use the Deque (which you created in assignment-1) for storing employee objects and manage them. So, if you haven't completed assignment-1 you won't be able to do assignment-2.

Modify the Deque in assignment-1 and make it a **EDLL** (Extended Doubly Linked List) by adding the following two functionalities to it:

- Adding an element from at the middle of the DLL (based on index/position passed).
- Removing an element from middle of the DLL (based on index/position passed).

XYZ Corporation has 3 types of employees:

- **Full-time**: Full time employees.
- **Contractors**: Part time employees from external agencies.
- **Interns**: College Interns (hired for 6 months internship).

Employees of XYZ corporation can have 3 types of status:

- **Active**: An active employee of the company
- **Inactive**: Is an employee of the company but currently inactive (LOA/Long Leave etc.).
- **Resigned**: Not an employee of the company anymore.

Create an employee manager (**XyzEmployeeManager**) which does the following:

- Creates an employee object.
    - First randomly get an employee type (Full-Time/Contractor/Intern) and then based on that create respective object by filling random values for member data like – {gender, status, Date of Birth, Date of Joining, Date of Leaving, no of leaves availed}.
    - If the type is a **Full-Time** employee then randomly assign a value for "no of leaves availed" between 0 to 22. (MAX number of leaves for a Full-Time employee is 22).
    - If the type is **Contractor** then randomly assign an agency.
    - If the type is **Intern** then randomly assign a college and branch.
    - For any employee if the status is randomly assigned as **Resigned** then the employee object should only have {Name, ID, Type, Status, Gender, DOB, DOJ, Date of Leaving}. For these types of employees create a separate deque of 'resigned employees'.
    - Based on the Type of employee and date-of-joining the company the date-of-leaving will be computed as : Full-Time("NA"), Contractor (Joining Date + 1 year), Intern (Joining Date + 6 months).
- Maintain the list of all the employees using a Deque (created in assignment-1). You might need to create 2 deques – 1 for Active and Inactive employees another one for resigned employees. Both these deques will be part of the Employee Manager class.

- Add a new **Full-Time** Employee
- Add a new **Contractor**
- Add a new **Intern**

HINT: You can use these 'Add' methods in the random creation of employee objects as well.

- Remove a Full-Time Employee
- Remove a Contractor
- Remove an Intern
  Once the employee is removed (after they have resigned), then move that employee to the “resigned employee” deque and only maintain minimal information required as mentioned above. Populate **date-of-leaving** the company for these types of employees.
- Print list of all the Full-Time employee’s summary.
- Print list of all the Contractors summary.
- Print list of all the Interns summary.
- Print the list of all the resigned employee’s summary.
- Print list of all the employee’s summary.
- Search an employee with name / employee ID.
- Make an Intern/Contractor as Full-Time employee.
- Add additional number (n) of leaves to all the Full-Time employees.
- Search for an employee (based on ID/Name).
- <Here, think of any other employee management scenarios … and implement them>

Employee ID Conventions: (Store it as a string)

- All the employee IDs should start with “XYZ”
- Followed by 4-digit numeric code eg:”0001”
- Finally appended with the employee type code: F(Full-Time), C(Contractor), I(Intern). e.g.: “XYZ0004C”

HINT: The requirement is to maintain all the employee’s list using a modified deque that we created in the beginning. However, it’s not mandatory to have a single deque to hold all the employees. You can come up with your own innovative ways of storing the employee objects. Separate deques for each type of employee and maintaining the head pointers of all the deques in an array in the manager class, maintaining 3 different pointers for different deques, OR maintaining everything in a single deque and then have class implementations does the rest … your choice. Come up with some good ideas here for effectively handling the memory and performance.

```
Sample framework for printing employees summary:
-------------------------------------------------------------------------------------------------------
| Employee Name              | ID       | Type   | Status | Gender | Date of Birth | Date Of Joining | ... |
-------------------------------------------------------------------------------------------------------
|                            |          |        |        |        |               |                 | ... |
|                            |          |        |        |        |               |                 | ... |
|                            |          |        |        |        |               |                 | ... |
|                            |          |        |        |        |               |                 | ... |
|                            |          |        |        |        |               |                 | ... |
|                            |          |        |        |        |               |                 | ... |
-------------------------------------------------------------------------------------------------------
```

Create an **XyzEmployeeIF** *Interface* which would support the following operations:

- Get any employee details like {name, ID, gender, DOB, …}. Add getter methods.
- Except {Name, ID, Gender & DOB} other fields should be updatable. Add only relevant setter methods.
- Print Employee details
  NOTE : For printing employee details make a note of the following constraints:
  - Common details are Name, ID, Type, Status, Gender, DOB and DOJ – these needs to be printed for all the employees.
  - For Full-Time employees additionally print leaves details (availed and left).
  - For Contractors additionally print external agency details.
  - For Interns additionally print college and branch details.

HINT: Use individual derived class names as : **XyzFullTimeEmployee**, **XyzContractorEmployee**, **XyzInternEmloyee**

```
A Sample print of 'Full-Time' Employee:
    Employee Name   : Kalvakuntla Chandrashekar Rao
    Employee ID     : XYZ00001F
    Employee Type   : Full-Time
    Employee Status : Active
    Gender          : Male
    Date of Birth   : 17 February 1954
    Date of Joining : 2 June 2004
    Leaves Availed  : 10
    Leaves Left     : 12
```

```
A Sample print of 'Contractor' Employee:
    Employee Name   : Iron Man
    Employee ID     : XYZ0001C
    Employee Type   : Contractor
    Employee Status : Active
    Gender          : Male
    Date of Birth   : 1 March 1963
    Date of Joining : 1 May 2008
    External Agency : Avengers
```

```
A Sample print of 'Intern' Employee:
    Employee Name   : Kriti Sanon
    Employee ID     : XYZ00001I
    Employee Type   : Intern
    Employee Status : Active
    Gender          : Female
    Date of Birth   : 27 July 1990
    Date of Joining : 10 January 2014
    College         : NIT Tiruchi
    Branch          : ECE
```

Below mentioned are all the supported types and values for implementation,

XYZ corporation has only 3 supported employee status:

- ACTIVE
- INACTIVE
- RESIGNED

XYZ corporation hires contractors only from the following external agencies:

- Avengers
- Justice League
- X-Men

XYZ corporation hires interns only from the following colleges:

- IIT Delhi
- IIT Mumbai

- IIT Kanpur
- IIT Hyderabad
- NIT Warangal
- NIT Tiruchi
- IIIT Hyderabad

XYZ corporation hires interns only from the following branches:

- CSE
- CSIT
- ECE

HINT: Use enums for all the above types and values.

Testing the Code:

Implement a menu-driven approach for the following as follows:

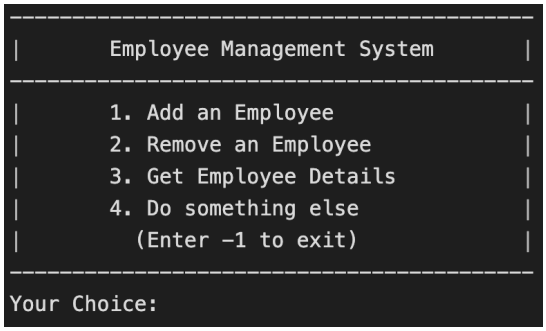
Main Menu Options	**Operation**	**Inputs from user**
1	Add an Employee	NA
2	Remove an Employee	Emp ID
3	Employee Details	NA
4	Others	NA

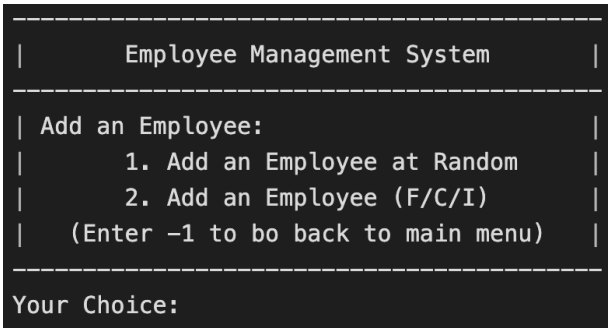
Sub-Menu-1 Options	**Add an Employee**	**Inputs from user**
1	Add an Employee at Random	N/A
2	Add an Employee (F/C/I)	Type

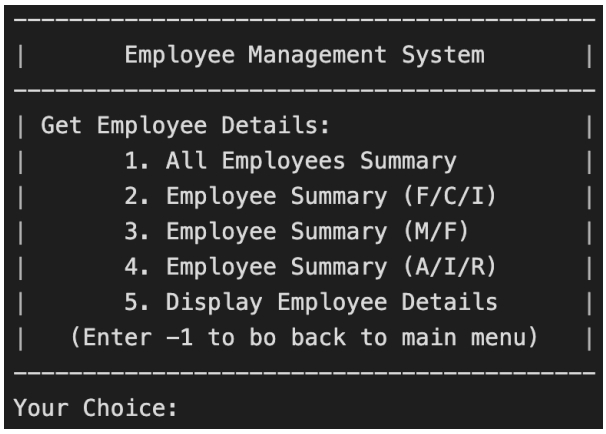
Sub-Menu-2 Options	**Employee Details**	**Inputs from user**
1	All Employees Summary	NA
2	Employee Summary (F/C/I)	Type
3	Employee Summary (M/F)	Gender
4	Employee Summary (A/I/R)	Status
5	Display Employee Details	Emp ID

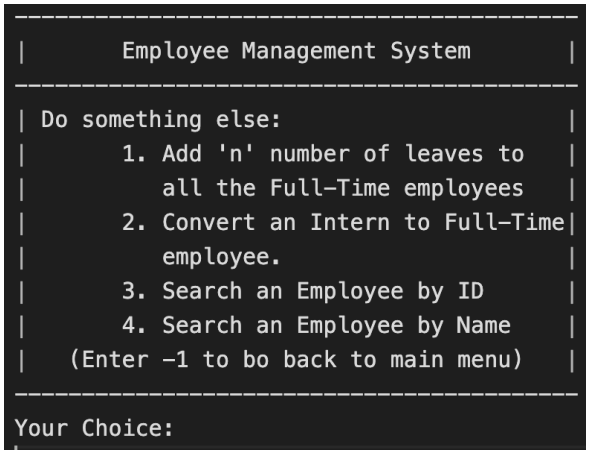
Sub-Menu-3 Options	**Others**	**Inputs from user**
1	Add ‘n’ number of leaves to all the Full-Time employees	NA
2	Convert an Intern to Full-Time employee.	Emp ID
3	Search an Employee by ID	Emp ID
4	Search an Employee by Name	Name / part of the name

Below given are the screen shots how the menus should look.
Once the choice is entered by the user … based on the above tables take necessary inputs from the user.









Important Guidelines for the Implementation:

- Use only C++ as programming language.
- DO NOT use any of the C++ STL libraries like vectors/maps/queue etc. You need to implement your own data structures and handle the memory management yourself.
- Follow PHYP Coding guidelines for naming conventions and others coding details.
- Write appropriate test cases to test your code. Try to see if you can add some features to the Employee manager. Come up with some new ideas here.
- PLAGIARISM is strictly prohibited and online solutions from websites like stack-exchange etc, will not be tolerated. I can figure out if the solution is copied from somewhere. So, avoid such short-cuts and work on the problem on your own.
- Avoid using online C++ compilers. Install "command-line-tools" in your macbook (which will install g++ compiler) and use it.
- Ask for help if needed.

Evaluation Parameters:

- How are you designing the solution.
- What all the OOPS concepts you are using.
- Any new thoughts/ideas implemented.
- What design patterns you are using.
- Memory management.
- Performance.