DSA LAB QUESTION

Instructions:

1. The interface template and the sample input file is provided.
2. You need to use concepts of OOPS to solve the problem.
3. Don't change the naming of the functions and the classes. It can fail the unit tests which are checking the correctness of the code. The test cases will be tested on the functions of the abstract class.

Problem Statement:
You are given a file which contains the information about students. You need to create separate linked lists for each of the information as shown in the implementation description.

Input format:
The input format is a CSV file which contains the **Name, Roll Number, Department, Course, Hostel and Club** information about the students.

Implementation:
You are given a interface file (.py, .cpp or .java), which you need to extend you logic using concepts of OOPS, and implement linked lists according to the details mentioned below:

1. There is a Position class, which is an abstract class, which contains the following abstract methods:
    a. Next(): it should return the next pointer to the element.
    b. element(): it should return the element which you are trying to access.
2. The Node class is a derived class of Position class, which should point to StudentRecord, which is invoked.
3. The Element class is an abstract class, which is a parent class to the StudentRecord class. It has the following variables:
    a. Name: it is used to denote the name of the element
4. The Student Record class is a derived class for the Element Class. It contains the following information:
    a. studentName: it stores the student name
    b. RollNumber: it store the student roll Number
5. The Entity class is an abstract class, which is a parent class for the linked list class. It has the following variables:
    a. Name: it stores the name of the entity
    b. Iterator: it returns the iterator to traverse the whole entity.
6. The linked list class is a derived class of the Entity class. You need to implement the functions inside the class as per your logic to complete the questions.

7. There is an global array for studentsRecords, which can be used to store the students record in the memory.
8. There is a global array declared for the Entity, which can be used to store all the linked lists, so that they can be referred later.

First, separate linked lists need to be created for different Hostels, Departments, Courses and Clubs, i.e., for each entity. The nodes in these linked lists need not to contain the actual copies of student data, but the pointer to the record of the student data in the memory, which can help to remove the problem of storing the duplicates in the memory. When a student record is added to the linked list, it is the pointer that points to the student record which is added to the list. Similar case with the deletion, the pointer is deleted from the list, but not the actual student record.

Output Format:
You need to solve the questions, and store it in a text file. Then you can run it against the unit test cases provided to check the correctness of the code.

Naming Conventions
File Naming Conventions:
1) Use lowercase letters for filenames. Must be student_solution.extension
2) The file names must have the appropriate extension.
   For python files .py, For C++ .cpp and for Java .java.

Class Name Conventions:
1) The class name must be StudentPortfolio that must inherit the provided interfaces.
2) Start Class names with a capital letter
3) The objects names must be in a fashion like for student entity s1, s2, s3, similarly for course c1, c2, c3 and so on.

Function Name Conventions:
1) Use lowercase letters for function and method names.

Questions:
1. For the given course, print the name of the students, along with their Roll Number.
2. Print the name of the students, who reside in the given hostel.
3. Print the name of the students, who are associated with the given club.
4. Delete the record of the given student from the given entity if present.
5. Add the new entry of the given student, into the given entity.