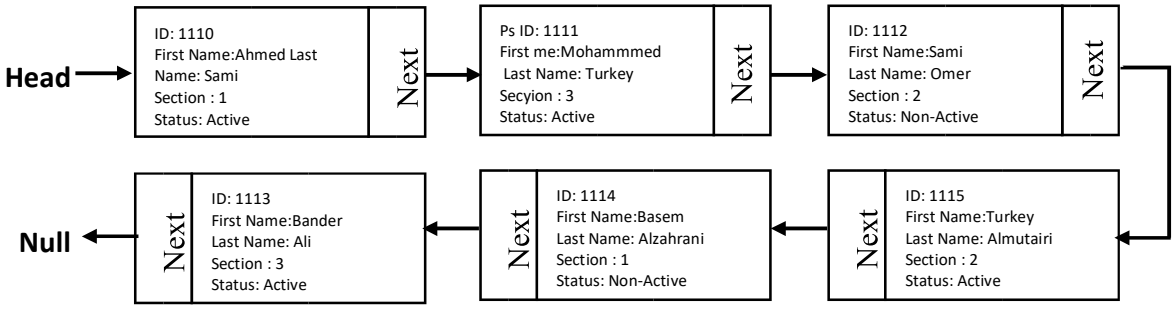
Concept Part

Question 1: (20 points)

1. **Concept Application:** Suppose a linked list contains the following information about the students. Write the steps to convert this single liked list into the circular single linked list. Your answer must indicate:
   - The **position of the pointer** and
   - The status of the **linked list** after every step.

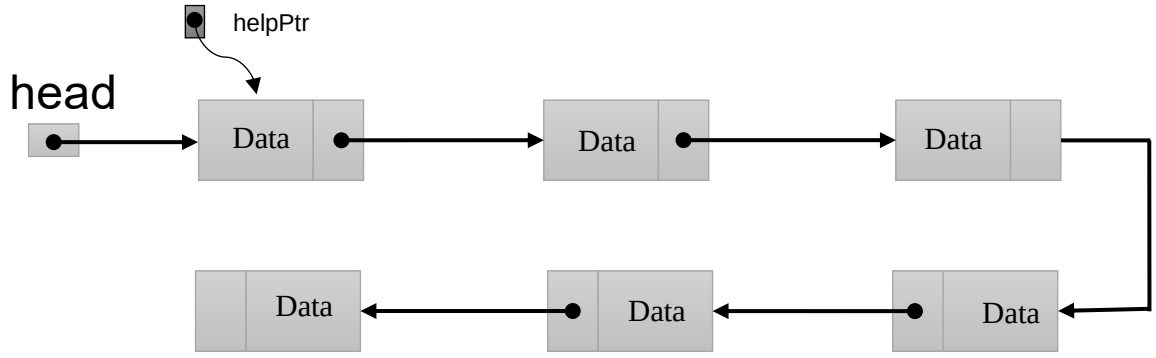


****Convert a single linked list into circular****

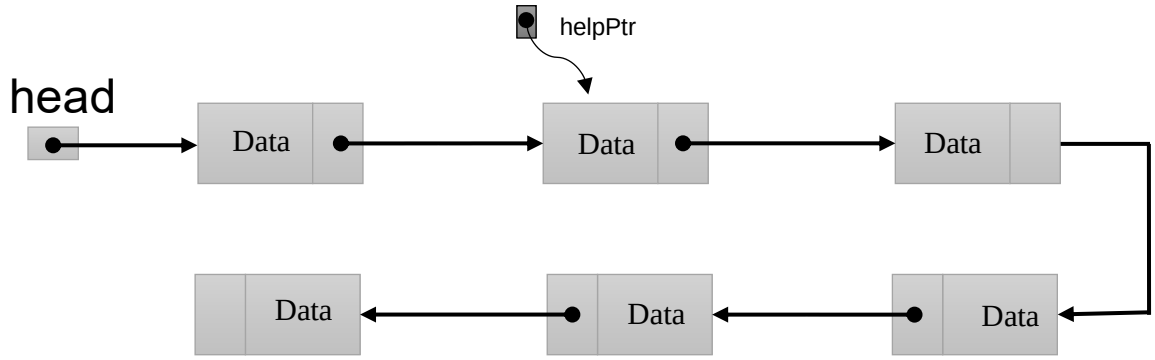
```
// Create a variable that point to the head of the single linked list
LLnode helpPtr = studentList.getHead();

// As long as the pointer.next is not pointing to null, it is still a single linked list
    while (helpPtr.getNext() != null){

        // The pointer will move to the next node
        helpPtr = helpPtr.getNext();

      }
        // If  the pointer.next reached to null, pointer.next will point to the head of the linked list
        // The linked list is converted to a circular list
        helpPtr.setNext(studentList.getHead());

    }
```

- The help pointer will take the value of head.
- At each iteration when helpPtr.getNext() doesn't equal null, it will move to the next value.
- Last iteration when helpPtr.getNext() reach to the null, its value is going to change to the head of linked list.

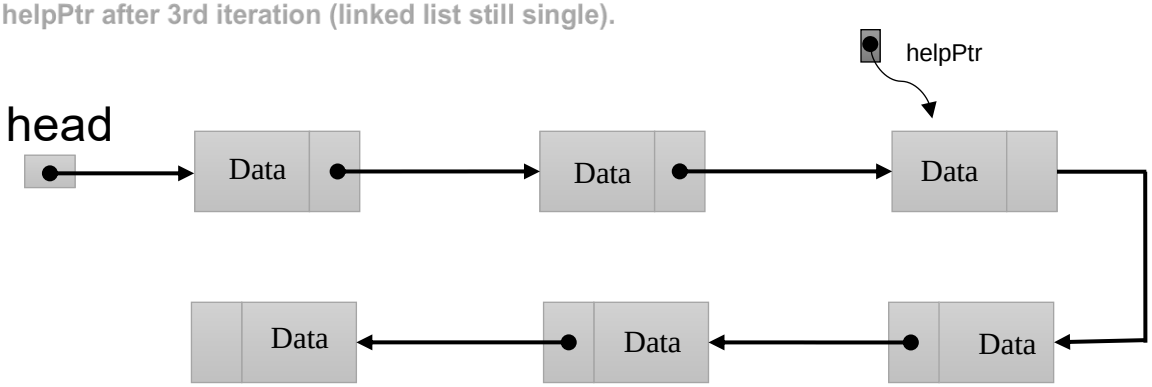
helpPtr after 1st iteration (linked list still single).



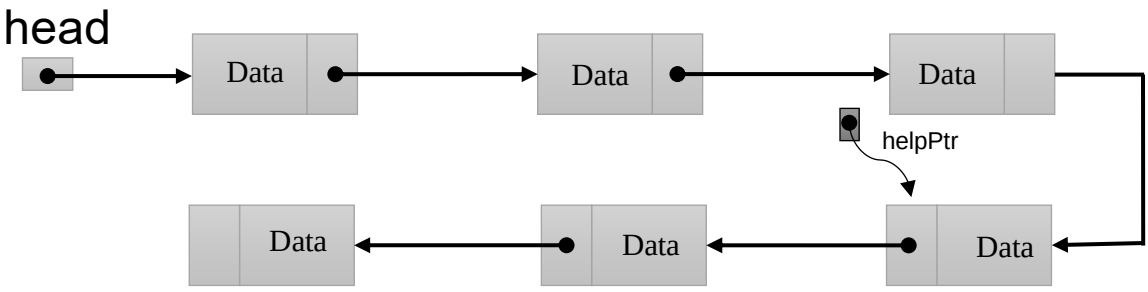
helpPtr after 2nd iteration (linked list still single).



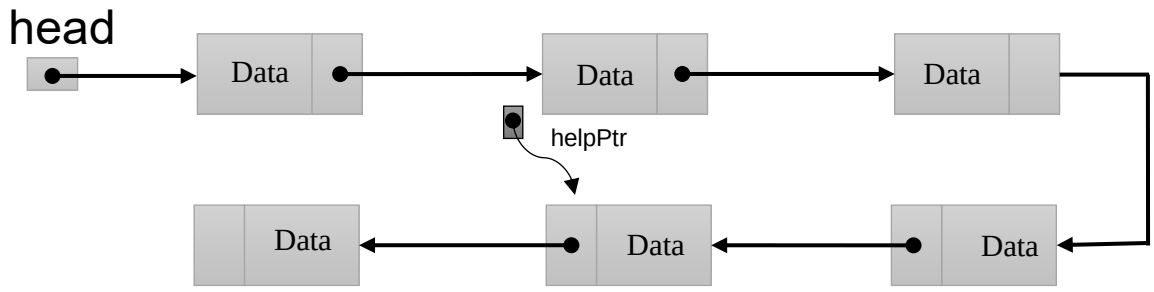
helpPtr after 3rd iteration (linked list still single).



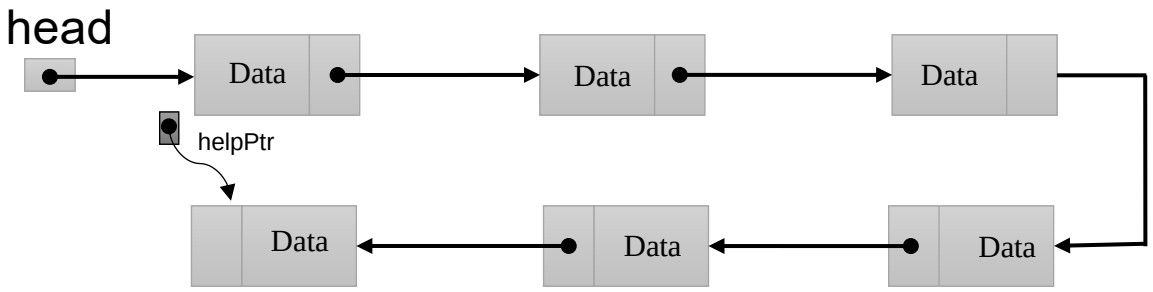
helpPtr after 4th iteration (linked list still single).



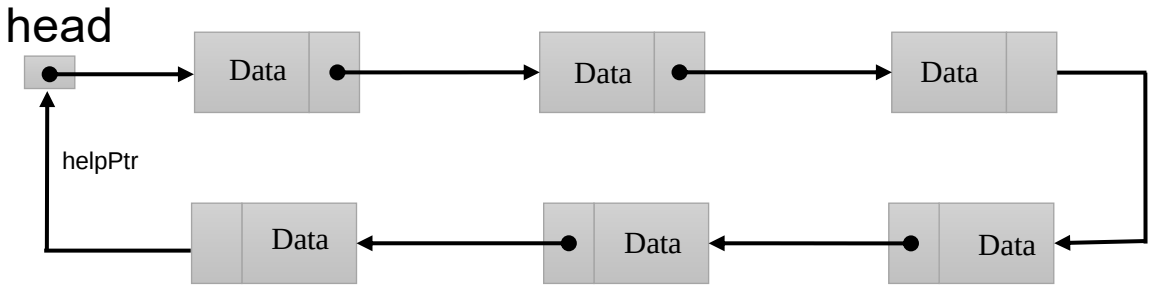
helpPtr after 5th iteration (linked list still single).



helpPtr after 5th iteration (linked list still single).



helpPtr after 6th iteration (Circular linked list).

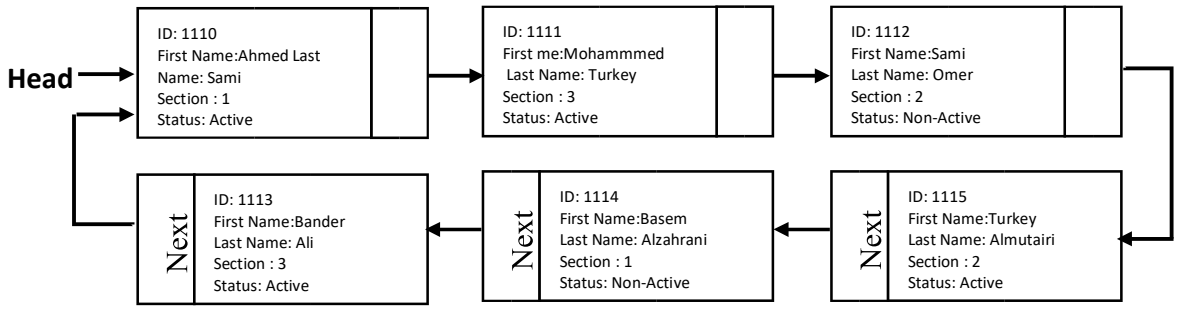


Question 2:(20 points)

2. Algorithm Write up: Suppose a **circular linked list** contains all students' information for all sections. The academic affairs employee wants to find all non active students in any required section.

Write an algorithm that searches the linked list for non active students in any section required by the academic affairs employee.

For example, an employee of academic affairs would like to display all non active students from section 2. If the linked list contains the following nodes



The output of the algorithm should print

```
The non active students in section 2 are:

1: ID:1112, Name: Sami Omer.
```

Algorithm
Input:

Number of a section

Output:

Non-Active students in the specified section

Algorithm:

- Input the number of a section
- Search for non-active students in a specified section
- Print non-active student

Pseudocode:

Step1: Input SECTION_NUMBER
Step2: COUNTER= 1
POINTER= HEAD
Step3: While (POINTER.NEXT != HEAD)
Step4: If (POINTER.SECTION= SECTION_NUMBER)
And if (POINTER= Non-Active)
Step5: THEN print non active student information

Method:
****Searches the linked list for non active students in any section required****

```
public static void displayNonActive(LinkedList studentList, Scanner input) {

    // Academic affairs employee enters the section required
    System.out.println("Enter section number: ");
    int section = input.nextInt();

    // Count students ranks
    int counter = 1;

    // Create a variable that point to the head of the single linked list
    LLnode helpPtr = studentList.getHead();

    // As long as the pointer.next not equal to the circular linked lists head
    while (helpPtr.getNext() != studentList.getHead()) {

        // If the pointer.section equals to the section that entered by academic affairs employee
        // And if the pointer.status is not Active (pointer.status==”Non-Active”)
        if (helpPtr.getSection() == section && helpPtr.getStatus() != “Active”) {


            // Print non active students information
            System.out.println("The non active students in section " + section + " are:");
            System.out.println(counter + ": ID:" + helpPtr.getID() + ", Name: " +
            helpPtr.getFirstName() + " " + helpPtr.getLastName());

            // Increase the counter of students rank
            counter++;
        }

        // The pointer will move to the next node
        helpPtr = helpPtr.getNext();
    }
}
```