**Question 1:**

1. Consider the operation with the following specification:

 **Void half(l: List)**

 **Requires**: List l is not empty and contains at least two elements

**Results:** Removes every other element from the list. For example, if l = [2, 13, 4, 5, 17, 8, 29] then this operation results l = [2, 4, 17, 29]. If l = [2, 4, 17, 29], then it results l = [2, 17].

Give implementation of this operation as a static method.

public static void half(List<int> L){

}

1. Using the operations in the specification of ADT List write a static method in the main class that inserts a new element in the list as the first element. Specification of the method is as follows:

**void insertFirst (List L, Type e)**

 **Requires:** The list L is not full.

**Results:** The new element is inserted as the first element in the list and is also made the current element.

public static void insertFirst(List<int> L, int e) {

}

**Question 2:** (Lists)

1. Add a new method/operation to the ADT List (Array Implementation) to find the previous element of the current. The method find\_prev(l : List), is specified below.

void findprev (l : List)

**Precondition/Requires:** The list l should not be empty or the current node should not be the first node.

**Results/Actions:** Sets the current to point to the node previous to the current node.

**Question 3:** (ADT)

Instructor of a certain course maintains records of the students attending the course, in a record keeping system. Each student’s record stores the information about a student: student’s name, id number, number of lectures attended, and the grades for two midterms and the final examination. At the start of the semester the instructor receives a *sorted* list of students for the course, from the registrar. During the semester, no new students are added to the list or deleted from it. The only operations, the instructor performs on the student’s records is to access them often to check or update the record.

(a) What is the *most suitable* adt for implementing the record keeping system and why?

(b) Give the specification for the adt.

(c) Give an implementation for the ADT.

**Question 4:** (Lists)

1. In the array based implementation of the ADT List what is the worst-case time complexity of insert operation algorithm. What is its best-case time complexity?
2. Answer the above question for the remove operation.
3. Answer parts (a) and (b) for linked implementation of the ADT list.

**Question 5:** (Lists)

Modify the ADT List (linked) implementation to add an operation/method length() that returns the number of elements in the list.

**Question 6:** Write the method removeOdd, member of the class LinkedList, which removes all the elements at odd positions (1, 3, 5…) from the list. The position of the first element of the list is 0. (When writing this method do not use any auxiliary data structure and do not call any method of the class LinkedList)

Example: L

head

: L after calling the method

head

public class LinkedList<T>{

 private Node<T> head;

private Node<T> current;

public LinkedList ()

{

head = current = null;

}

public void removeOdd() {

}

**Question 7:** Write the method swap, a member of the class LinkedList, which takes as input two integers i and j. The method swaps the two elements at position i and j. The position of the first element is 0, and the list must contain at least j+1 elements.

public void swap(int i, int j) {

}

**Question 8**: Consider the following linked list l:

head

Redraw the list after each of the following groups of instructions:

1. l.findFirst(); l.insert("c");

|  |
| --- |
|  |

1. l.findNext(); l.insert("b");

|  |
| --- |
|  |

1. l.findNext(); l.findNext(); l.remove();

|  |
| --- |
|  |

1. l.remove();

|  |
| --- |
|  |