Homework 1

CSE 274
Deadline 09/10/2022

Note: The focus of this homework is on singly Linked List. Accordingly, in this homework a linked list refers to a singly linked list.
Note: For this assignment a single `Application.java` file should be submitted. At the top part of the file, write the answers to questions 1 to 5 as Java comments using the operator `//`. The `Application.java` file should include the requested method in question 7 and should be executable to produce the correct output.

For questions 1 to 5, consider a linked list with N number of links.

Question 1 (10 points): How many steps it takes to insert a new link at the beginning part of a linked list?

Question 2 (10 points): How many steps it takes to insert a new link at the ending part of a linked list?

Question 3 (10 points): In average, how many steps it takes to find a link that includes a specific key, i.e. `Data==key`?

Question 4 (10 points): Once a link list is created, does it have a fixed number of links?

Question 5 (10 points): Is there any relationship between the memory locations at which the links of a linked list are placed?

Question 6 (20 points): A programmer has been asked to develop a `delete` method for the `LinkList` class that we worked on in Lab 1 session:

```
public void delete(int key)
```

The above method receives a `key` as an input, searches the linked list to find a link with `iData==key`, and deletes the link from the linked list. The programmer has proposed the following logic for developing the `delete` method:

- Scenario 1: If the linked list is empty, `delete` method should `return`.
- Scenario 2: If the linked list has only one link, there are two possible conditions:
  - Condition 1: If the first link does have the key, the first link should be deleted and then `delete` method should `return`.
  - Condition 2: If the first link does not have the key, the `delete` method should `return`.
- Scenario 3: If the link list has more than one link, there are two possibilities:
  - Condition 1: If the first link has the key, the first link should be deleted and `delete` method should `return`.
  - Condition 2: If the first link does not have the key, `previous` should be set as a reference to the first link, `current` should be set as a reference to the second link, and using a `while` loop a search should be initiated to find the link with `Data==key`.

The developed method is provided in `Application.java` file. To check the developed method, we perform the following tests:

Test 1: Calling `delete` method on an empty linked list:

Erase the body of `main` method, copy the following code in the body of `main` method, and check if correct output is being produced?

```
LinkList myLinkedlist = new LinkList();
System.out.println("Linked list before calling delete method:" );
myLinkedlist.displayList();

myLinkedlist.delete(9);
System.out.println("Linked list after calling delete method:" );
myLinkedlist.displayList();
```

Test 2: Calling `delete` method on a linked list with one link which has the `key`:

Erase the body of `main` method, copy the following code in the body of `main` method, and check if correct output is being produced?

```
LinkList myLinkedlist = new LinkList();
myLinkedlist.insertFirst(9, 10.3);
System.out.println("Linked list before calling delete method:" );
myLinkedlist.displayList();

myLinkedlist.delete(9);
System.out.println("Linked list after calling delete method:" );
myLinkedlist.displayList();
```

Test 3: Calling `delete` method on a linked list with one link which does not have the `key`:

Erase the body of `main` method, copy the following code in the body of `main` method, and check if correct output is being produced?

```
LinkList myLinkedlist = new LinkList();
myLinkedlist.insertFirst(11, 10.3);
System.out.println("Linked list before calling delete method:" );
myLinkedlist.displayList();

myLinkedlist.delete(9);
System.out.println("Linked list after calling delete method:" );
myLinkedlist.displayList();
```

Test 4: Calling `delete` method on a linked list with more than one link where the first link has the `key`:

Erase the body of `main` method, copy the following code in the body of `main` method, and check if correct output is being produced?

```
LinkList myLinkedlist = new LinkList();
myLinkedlist.insertFirst(14, 10.3);
myLinkedlist.insertFirst(15, 10.3);
myLinkedlist.insertFirst(9, 10.3);

System.out.println("Linked list before calling delete method:" );
myLinkedlist.displayList();

myLinkedlist.delete(9);
System.out.println("Linked list after calling delete method:" );
myLinkedlist.displayList();
```

Test 5: Calling `delete` method on a linked list with more than one link where the first link does not have the `key`, but another link has the `key`.
Erase the body of `main` method, copy the following code in the body of `main` method, and check if correct output is being produced?

```
LinkList myLinkedlist = new LinkList();
myLinkedlist.insertFirst(15, 10.3);
myLinkedlist.insertFirst(9, 10.3);
myLinkedlist.insertFirst(16, 10.3);

System.out.println("Linked list before calling delete method:" );
myLinkedlist.displayList();

myLinkedlist.delete(9);
System.out.println("Linked list after calling delete method:" );
myLinkedlist.displayList();
```

Test 6: Calling `delete` method on a linked list with more than one link where no link has the `key`.
Erase the body of `main` method, copy the following code in the body of `main` method, and check if correct output is being produced?

```
LinkList myLinkedlist = new LinkList();
myLinkedlist.insertFirst(21, 10.3);
myLinkedlist.insertFirst(15, 10.3);
myLinkedlist.insertFirst(16, 10.3);

System.out.println("Linked list before calling delete method:" );
myLinkedlist.displayList();

myLinkedlist.delete(9);
System.out.println("Linked list after calling delete method:" );
myLinkedlist.displayList();
```

Considering the above tests, does the developed `delete` method work as expected? If your answer is "no" propose a test for which the `delete` method is not producing the correct output.

Question 7 (30 points): Develop the following method for the `LinkList` class:

```
public void insertOrdered(int id, double dd)
```

The above method takes an `int` input and a `double` input from the user, creates a new link with the given data, and inserts the link at an appropriate place in the linked list, so that the integer data in the links are in increasing order:

`iData` in the 1st link $\leq$ `iData` in the 2nd link $\leq$ `iData` in the 3rd link $\leq \cdots \leq$ `iData` in the last link.

Please notice that, for the above method to work properly new links should be added to the linked list using only the above method. To test the developed method, erase the body of `main` method, copy the following code in the body of `main` method, and check if correct output is being produced?

```
LinkList myLinkedlist = new LinkList();
myLinkedlist.insertOrdered(21, 10.3);
myLinkedlist.insertOrdered(15, 11.8);
myLinkedlist.insertOrdered(18, 12.1);
myLinkedlist.insertOrdered(9, 19.9);
myLinkedlist.insertOrdered(50, 15.7);
myLinkedlist.insertOrdered(26, 11.4);

myLinkedlist.displayList();
```

Note: Please keep the above lines of code in the body of `main` method when submitting your response to the homework.