LINKED LIST DATA STRUCTURE (TUTORIAL 3)

PART A

Question 1

Consider the below linked list

Write code segment to change the above linked list to the link list given below


```java
public class LinkList {
private Link first;

	public LinkList() {
		first = null;
	}//constructor

	public boolean isEmpty() {
		return (first == null);
	}//isEmpty method

	public void displayList() {
		Link current = first;

		while (current != null) {
			current.displayDetails();
			current = current.next;
		}

	}//displayList method

	public void insertFirst(String n) {
		Link l = new Link(n);
		l.next = first;
		first = l;
	}//insertFirst method

	public void swap(String s1, String s2) {
		Link current1 = first;

		while (current1.name != s1) {
			current1 = current1.next;
		}

		Link current2 = first;
		while (current2.name != s2) {
			current2 = current2.next;
		}

		String temp = current1.name;
		current1.name = current2.name;
		current2.name = temp;
	}
}
```

```java
public class Link {
	public String name;
	public Link next;

	public Link(String n) {
		this.name = n;
	}//constructor

	public void displayDetails() {
			System.out.print(name);
			System.out.print(" ");
	}//displayDetails method
}
```

```java
public class MainQ1 {

	public static void main(String[] args) {

		LinkList list = new LinkList();

		list.insertFirst("Nisha");
		list.insertFirst("Mihiri");
		list.insertFirst("Akikla");
		list.insertFirst("Mithila");
		list.displayList();

		list.swap("Mithila", "Mihiri");
		System.out.println("");//new line
		list.displayList();
	}

}
```

Question 2

Consider the link class and linked list class given below

Link
- int iData; - Link next;
- Link(int id) - void displayLink()

LinkList
- Link first;
- LinkList() - boolean isEmpty() - void displayList() - boolean delete(int key) - boolean insertAfter(int key, int newData) - Link find(int key)

i. Implement insertAfter(int key, int newData) method of the LinkList class. InsertAfter() method finds the link with the given key and the new link (with newData value) is inserted immediately after that

ii. Implement the delete(int key) method of the LinkList class. delete() method finds the link with the given key and remove it from the link list

iii. Write an application to enter numbers from the keyboard to a link list
   1. Add a new link after a given number and display the list
   2. Delete a link from the link list and display the list

```
public class LinkList {
	Link first;
	public LinkList() {
		first = null;
	}//constructor
	public boolean isEmpty() {
		return (first == null);
	}//isEmpty
	public void displayList() {
		Link current = first;
		while (current != null) {
			current.displayLink();
			current = current.next;
		}//while
	}//displayList
	public boolean delete(int key) {
		Link temp = find(--key);
		Link temp1;
		if (key == 0) {
			first = first.next;
		} else {
			temp1 = temp.next;
			temp.next = temp1.next;
			temp1 = null;
		} //if-else
		return true;
	}//delete
	public boolean insertAfter(int key, int newData) {
		Link newlink = new Link(newData);
		if (key == 0) {
			Link link = new Link(newData);
			link.next = first;
			first = link;
		} else {
			Link temp = find(key);
			newlink.next = temp.next;
			temp.next = newlink;
		}//if-else
		return true;
	}//insertAfter
	Link find(int key) {
		Link temp = first;
		for (int i = 1; i < key; i++) {
			temp = temp.next;
		}//for
		return temp;
	}//find
}//class
```

```
import java.util.Scanner;

public class Application {

	public static void main(String[] args) {
		LinkList l = new LinkList();
		Scanner sc = new Scanner(System.in);
		System.out.println("How many numbers? ");
		int k = sc.nextInt();

		for (int i = 0; i < k; i++) {
System.out.print("Enter number " + (i+1) + ": ");
		l.insertAfter(i, sc.nextInt());
		}
		l.displayList();
		System.out.println("");

		l.insertAfter(2, 200);
		l.displayList();
		System.out.println("");

		l.delete(2);
		l.displayList();
	}//main
}//class
```

```
public class Link {
	int iData;
	Link next;

	public Link(int id) {
		iData = id;
		next = null;
	}

	public void displayLink() {
		System.out.print(iData);
		System.out.print(" ");
	}

}//Link class
```

PART B

Question 3

How do you implement a "Stack" using a linked list instead of an array?
(use insertFirst for push & deleteFirst for pop, since in stack insertion and deletion both are done from top)

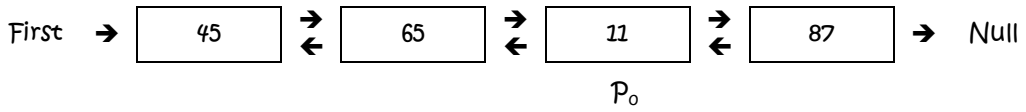
```
public class LinkedStack {
	private Link top;

	public LinkedStack() {
		top = null;
	}//constructor

	public void push(int i) {
		Link link = new Link(i);
		link.next = top;
		top = link;
	}//push

	public int pop() {
		if (isEmpty()) {
			System.out.println("Stack underflow");
			return 0;
		} else {
			Link temp = top;
			top = top.next;
			return temp.data;
		}

	}//pop

	public int peek() {
		int i = pop();
		push(i);
		return i;
	}//peek

	public boolean isEmpty() {
		return (top == null);
	}//isEmpty

}//class
```

```
public class Link {
	public int data;
	public Link next;

	public Link(int i) {
		data = i;
		next = null;
	}//constructor

}//class
```

```
public class Main {

	public static void main(String[] args) {
		LinkedStack stack = new LinkedStack();

		for (int i = 0; i < 5; i++) {
			stack.push(i);
		}//for

		System.out.println(stack.peek());

		while (!stack.isEmpty()) {
			System.out.println(stack.pop());
		}

	}//main

}//class
```

Question 4

i. Consider the following doubly link list and illustrate and write the steps to be followed, if the link P_o is to be deleted



ii. Illustrate the steps and write the statements to be followed to delete the first link

iii. Illustrate the steps and write the statements to be followed to insert a new link as the first link

```
public class DoublyLinkedList {
	private Link first, last;

	public DoublyLinkedList() {
		first = last = null;
	}//constructor

	public boolean isEmpty() {
		return (first == null);
	}//isEmpty

	public void displayList() {
		Link current = first;

		while (current != null) {
			current.displayDetails();
			current = current.next;
		}

	}//displayList

	public void insertFirst(int i) {
		Link newlink = new Link(i);

		if (isEmpty()) {
			first = last = newlink;
			first.previous = last.next = null;
		} else {
			first.previous = newlink;
			newlink.next = first;
			newlink.previous = null;
			first = newlink;
		}//if-else

	}//insertFirst

	public Link deleteFirst() {
		Link temp = first;
		first = first.next;
		first.previous = null;
		return temp;
	}//deleteFirst
```

```
package doublyLinkedList;

public class Link {
	public int data;
	public Link next, previous;

	public Link(int i) {
		data = i;
		next = previous = null;
	}//constructor

	public void displayDetails() {
		System.out.println(data + " ");
	}//displayDetails

}//class
```

```
	public void insertLast(int i) {
		Link newlink = new Link(i);
		Link temp = first;
		if (isEmpty()) {
			insertFirst(i);
		} else {
			while (temp.next != null) {
				temp = temp.next;
			}//while
			temp.next = newlink;
			newlink.previous = temp;
		}//if-else

	}//insertLast

	public Link deleteLast() {
		Link current = first;
		while (current.next.next != null) {
			current = current.next;
		}//while
		current.next = null;
		return current;
	}//deleteLast

	public void insertAfter(int i, int position) {
		Link newlink = new Link(i);
		if (isEmpty()) {
			first = last = newlink;
			first.previous = last.next = null;
		} else {
			Link current = first, temp = null;
			for (int j = 1; j < position; j++) {
				current = current.next;
			}//for

			temp = current.next;
			temp.previous = current;
			current.next = newlink;
			newlink.previous = current;
			newlink.next = temp;
			temp.previous = newlink;
		}//if-else

	}//insertAfter

	public Link deleteAfter(int position) {
		if (isEmpty()) {
			return null;
		} else if (position == 0) {
			return first.next;
		} else {
			Link current = first;
			for (int j = 1; j < position; j++) {
				current = current.next;
			}//for

			current.next = current.next.next;
			return current;
		}//if-else
	}//deleteAfter

}//class
```

```
package doublyLinkedList;
import java.util.Scanner;

public class Main {

public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
DoublyLinkedList dl = new DoublyLinkedList();

/*for (int i = 0; i < 4; i++) {
System.out.print("Enter number: ");
int number = sc.nextInt();
dl.insertFirst(number);
}
dl.displayList();
System.out.println("");

dl.deleteFirst();
dl.displayList();
System.out.println("");*/

/*for (int i = 0; i < 4; i++) {
System.out.print("Enter number: ");
int number = sc.nextInt();
dl.insertLast(number);
}
dl.displayList();
System.out.println("");

dl.deleteLast();
dl.displayList();
System.out.println("");*/

for (int i = 0; i < 5; i++) {
System.out.print("Enter number: ");
int number = sc.nextInt();
dl.insertLast(number);
}
dl.insertAfter(5, 1);
dl.displayList();
System.out.println("");

dl.deleteAfter(2);
dl.displayList();

}//main

}//class
```

EXTRA:

How do you implement a "Queue" using a linked list instead of an array?
(use insertLast for insert & deleteFirst for remove, since in queue insertion is done at rear end and deletion is done at front end)

```
public class LinkedQueue {
	private Link front;

	public LinkedQueue() {
		front = null;
	}//constructor

	public void insert(int i) {
		Link link = new Link(i);
		if (front == null) {
			front = link;
		} else {
			Link temp = front;

			while (temp.next != null) {
				temp = temp.next;
			}//while
			temp.next = link;
		}//if-else

	}//insert

	public int remove() {
		if (isEmpty()) {
			System.out.println("Stack underflow");
			return 0;
		} else {
			Link temp = front;
			front = front.next;		//delete
			return temp.data;
		}

	}//remove

	public int peekFront() {
		if (isEmpty()) {
			System.out.println("Stack underflow");
			return 0;
		} else {
			Link temp = front;
			return temp.data;		//without delete
		}
	}//peekFront

	public boolean isEmpty() {
		return (front == null);
	}//isEmpty

}//class
```

```
public class Main {

	public static void main(String[] args) {
		LinkedQueue queue = new LinkedQueue();

		for (int i = 0; i < 5; i++) {
			queue.insert(i);
		}//for
		System.out.println(queue.peekFront());

		while (!queue.isEmpty()) {
			System.out.println(queue.remove());
		}

	}//main

}//class
```

```
public class Link {
	public int data;
	public Link next;

	public Link(int i) {
		data = i;
		next = null;
	}//constructor

}//class
```