**WIA 1002**

**TUTORIAL 5 LinkedList and Doubly LL**

**Question 1**

1 public E xyz(int index,E e)

2 {

3 Node<E> current=head;

4 Node<E> temp;

5 if(index<0) return null;

6 else if(index>=size-1) {

7 this.addLast(e);

8 return null;

9 }

10 else if(index==0) {

11 temp=head;

12 head.element=e;

13 return temp.element;

14 }else{

15 for (int i = 1; i < index; i++) {

16 current=current.next;

17 }

18 temp=current.next;

19 current.next.element=e;

20 return temp.element;

21 }

**}**

Given method xyz with 2 arguments:

a) Based on the above source code, explain what the lines of code do from line 10 – 21.

b) What is the main purpose of the method xyz()?

c) Obviously, there are some bugs in the source code. Debug the code and make it concise and simpler.

Answer:

1. Line 10 to Line 21 adds the element e of generic type E into the linked list at the index of index. Line 10 to 14 adds e to the first index of the linked list, line 15-17 iterate forward beginning at index1 and stop at the requested index, line 18 copy current.next node to temp, line 19 assign e to element of current.next node, line 20-21 return the old value that has been replaced with (using temp node).
2. To set new value based on given index and to return old value that has been replaced with.
3. public E xyz(int index, E e){

Node<E> current= head;

Node<E> temp = new Node<>(e);

if(index <0 || index > size)

return null;

if(index == 0)

temp.next=head;

head=temp;

if(index >= size)

addLast(e);

else{

for(int i=0;i<index-1;i++){

current = current.next; //traverse nodes

}

temp.next=current.next;

current.next=temp;

size++;

return temp.element;

}

**Question 2**

1 else{

2 Node<E> temp = head;

3 for(int i=0; i<index; i++){

4 temp = temp.next;

5 }

6 element = temp.element;

7 temp.next.prev = temp.prev;

8 temp.prev.next = temp.next;

9 temp.next = null;

10 temp.prev = null;

11 size --;

12 }

Based on the source code above, assume the index given is 3

a) Explain what the lines of code do from line 2-11.

b) Draw the nodes for lines 7 – 10

1. At line 2, we copy the head to the temp node, at line 3 to line 5 we traverse the node to the index index, at line 6 we copy the element in the node of index index to element, then at line 7 to line 10 we link the previous node and next node of the node of index index together and then disconnect the node at index index to both nodes. Then finally at line 11 we reduce the size property of our doubly linked list after the removal of the node.

Node condition at Line 7 and 8:

┌─────────────────────┐

head: V index: │ tail:

┌──────┐ ┌──────┐ ┌──────┐ ┌──────┐ ┌──────┐

│ prev │<──│ prev │<──│ prev │<──│ prev │<──│ prev │

├──────┤ ├──────┤ ├──────┤ ├──────┤ ├──────┤

│ el 0 │...│ el 1 │ │ el 2 │ │ el 3 │...│ el 4 │

├──────┤ ├──────┤ ├──────┤ ├──────┤ ├──────┤

│ next │──>│ next │──>│ next │──>│ next │──>│ next │

└──────┘ └──────┘ └──────┘ └──────┘ └──────┘

│ ^

└─────────────────────┘

Node condition at Line 9 and 10:

┌─────────────────────┐

head: V index: │ tail:

┌──────┐ ┌──────┐ ┌──────┐ ┌──────┐ ┌──────┐

│ prev │<──│ prev │ │ prev │ │ prev │<──│ prev │

├──────┤ ├──────┤ ├──────┤ ├──────┤ ├──────┤

│ el 0 │...│ el 1 │ │ el 2 │ │ el 3 │...│ el 4 │

├──────┤ ├──────┤ ├──────┤ ├──────┤ ├──────┤

│ next │──>│ next │ │ next │ │ next │──>│ next │

└──────┘ └──────┘ └──────┘ └──────┘ └──────┘

│ ^

└─────────────────────┘

Post condition:

head: tail:

┌──────┐ ┌──────┐ ┌──────┐ ┌──────┐

│ prev │<──│ prev │<──│ prev │<──│ prev │

├──────┤ ├──────┤ ├──────┤ ├──────┤

│ el 0 │...│ el 1 │ │ el 3 │...│ el 4 │

├──────┤ ├──────┤ ├──────┤ ├──────┤

│ next │──>│ next │──>│ next │──>│ next │

└──────┘ └──────┘ └──────┘ └──────┘

**Question 3**

A doubly linked list keeps a set of characters. The head, the middle and the tail nodes respectively

contains alphabet ‘a’, ‘b’ and ‘c’. These nodes are in successive order. Create a new node that contains alphabet ‘z’. Add this new node at the last location of this linked list. Draw all of these nodes including their correct references.

head: mid: tail:

┌──────┐ ┌──────┐ ┌──────┐ ┌──────┐

│ prev │<──│ prev │<──│ prev │<──│ prev │

├──────┤ ├──────┤ ├──────┤ ├──────┤

│ 'a' │ │ 'b' │ │ 'c' │ │ 'z' │

├──────┤ ├──────┤ ├──────┤ ├──────┤

│ next │──>│ next │──>│ next │──>│ next │

└──────┘ └──────┘ └──────┘ └──────┘

//don’t know need code or not

public void addLast(E z){

Node<E> = new Node<element, null, tail);

if(tail !=null){

tail.next = temp;

}

tail=temp;

if(head == null){

head = temp;

}

size++;

System.out.println(“adding: “ + z);

}