

Data Structures & Algorithms (CSE2001) Lab-4

KHAN MOHD OWAIS RAZA (20BCD7138)

Q.1] Write a program to implement doubly linked list and its operations.

```
/**
Name: KHAN MOHD OWAIS RAZA
ID : 20BCD7138
Course: Data Structures & Algorithm
Code: CSE2001
Slot: L19+L20
**/
/* Lab-4 (01-10-2022)*/
/* Java code to implement doubly linked list and its operations */
package CSE2001_Lab4_20BCD7138;
import java.util.Scanner;
class Node{
protected int data;
protected Node next, prev;
public Node(){
next = null;
prev = null;
data = 0;
}
public Node(int d, Node n, Node p){
data = d;
next = n;
prev = p;
}
public void setLinkNext(Node n){
next = n;
}
public void setLinkPrev(Node p){
prev = p;
}
public Node getLinkNext(){
return next;
}
public Node getLinkPrev(){
return prev;
}
public void setData(int d){
data = d;
}
public int getData(){
return data;
}}
class DLL{
protected Node start;
protected Node end ;
public int size;
```

```

public DLL(){
    start = null;
    end = null;
    size = 0;
}
public boolean isEmpty(){
    return start == null;
}
public int getSize(){
    return size;
}
public void insertAtStart(int val){
    Node nptr = new Node(val, null, null);
    if(start == null){
        start = nptr;
        end = start;
    }
    else{
        start.setLinkPrev(nptr);
        nptr.setLinkNext(start);
        start = nptr;
    }
    size++;
}
public void insertAtEnd(int val){
    Node nptr = new Node(val, null, null);
    if(start == null){
        start = nptr;
        end = start;
    }
    else{
        nptr.setLinkPrev(end);
        end.setLinkNext(nptr);
        end = nptr;
    }
    size++;
}
public void insertAtPos(int val , int pos){
    Node nptr = new Node(val, null, null);
    if (pos == 1){
        insertAtStart(val);
        return;
    }
    Node ptr = start;
    for (int i = 2; i <= size; i++){
        if (i == pos){
            Node tmp = ptr.getLinkNext();
            ptr.setLinkNext(nptr);
            nptr.setLinkPrev(ptr);
            nptr.setLinkNext(tmp);
            tmp.setLinkPrev(nptr);
        }
        ptr = ptr.getLinkNext();
    }
}

```

```

size++ ;
}
public void deleteAtPos(int pos){
if (pos == 1){
if (size == 1){
start = null;
end = null;
size = 0;
return;
}
start = start.getLinkNext();
start.setLinkPrev(null);
size--;
return ;
}
if (pos == size){
end = end.getLinkPrev();
end.setLinkNext(null);
size-- ;
}
Node ptr = start.getLinkNext();
for (int i = 2; i <= size; i++){
if (i == pos)
{
Node p = ptr.getLinkPrev();
Node n = ptr.getLinkNext();
p.setLinkNext(n);
n.setLinkPrev(p);
size-- ;
return;
}
ptr = ptr.getLinkNext();
}}
public void display(){
System.out.print("\nDoubly linked list: ");
if (size == 0){
System.out.print("Empty\n");
return;
}
if (start.getLinkNext() == null) {
System.out.println(start.getData() );
return;
}
Node ptr = start;
System.out.print(start.getData()+ " <-> ");
ptr = start.getLinkNext();
while (ptr.getLinkNext() != null)
{
System.out.print(ptr.getData()+ " <-> ");
ptr = ptr.getLinkNext();
}
System.out.print(ptr.getData()+ "\n");
}}
public class Question1{

```

```

public static void main(String[] args){
Scanner scan = new Scanner(System.in);
DLL list = new DLL();
System.out.print("\nKHAN MOHD OWAIS RAZA (20BCD7138)");
System.out.print("\nCSE2001 Lab-4");
System.out.print("\n");
System.out.print("\n");
System.out.print("\nJava code to implement doubly linked list and its
operations");
System.out.print("\n");
char ch;
do{
System.out.println("\nSelect an operation: \n");
System.out.println("[1] Insert element at beginning");
System.out.println("[2] Insert element at ending");
System.out.println("[3] Insert element at a position");
System.out.println("[4] Delete element from a position");
System.out.println("[5] Check if the list is empty");
System.out.println("[6] Display size");
int choice = scan.nextInt();
switch (choice){
case 1 : System.out.println("\nEnter the element to be inserted: ");
list.insertAtStart( scan.nextInt() );
break;
case 2 : System.out.println("\nEnter the element to be inserted: ");
list.insertAtEnd( scan.nextInt() );
break;
case 3 : System.out.println("\nEnter the element to be inserted: ");
int num = scan.nextInt() ;
System.out.println("Enter the position: ");
int pos = scan.nextInt() ;
if (pos < 1 || pos > list.getSize()) System.out.println("Invalid
position!!\n");
else list.insertAtPos(num, pos);
break;
case 4 : System.out.println("Enter the position: ");
int p = scan.nextInt();
if (p < 1 || p > list.getSize()) System.out.println("Invalid
position!!\n");
else list.deleteAtPos(p);
break;
case 5 : System.out.println("Empty status: " + list.isEmpty());
break;
case 6 :
System.out.println("Size: " + list.getSize() + " \n");
break;
default : System.out.println("Incorrect entry!! \n ");
break;
}
list.display();
System.out.println("\nPress 'C' to continue and 'S' to stop\n");
ch = scan.next().charAt(0);
} while (ch == 'C');
}}

```

```
@ Javadoc Declaration Console × Coverage
<terminated> Question1 (1) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe

KHAN MOHD OWAIS RAZA (20BCD7138)
CSE2001 Lab-4

Java code to implement doubly linked list and its operations

Select an operation:

[1] Insert element at beginning
[2] Insert element at ending
[3] Insert element at a position
[4] Delete element from a position
[5] Check if the list is empty
[6] Display size
1

Enter the element to be inserted:
10

Doubly linked list: 10

Press 'C' to continue and 'S' to stop
C

Select an operation:

[1] Insert element at beginning
[2] Insert element at ending
[3] Insert element at a position
[4] Delete element from a position
[5] Check if the list is empty
[6] Display size
2

Enter the element to be inserted:
20

Doubly linked list: 10 <-> 20

Press 'C' to continue and 'S' to stop
C

Select an operation:

[1] Insert element at beginning
[2] Insert element at ending
[3] Insert element at a position
[4] Delete element from a position
[5] Check if the list is empty
[6] Display size
2

Enter the element to be inserted:
40

Doubly linked list: 10 <-> 20 <-> 30 <-> 40

Press 'C' to continue and 'S' to stop
C
```

Select an operation:

- [1] Insert element at beginning
- [2] Insert element at ending
- [3] Insert element at a position
- [4] Delete element from a position
- [5] Check if the list is empty
- [6] Display size

2

Enter the element to be inserted:

50

Doubly linked list: 10 <-> 20 <-> 30 <-> 40 <-> 50

Press 'C' to continue and 'S' to stop

C

Select an operation:

- [1] Insert element at beginning
- [2] Insert element at ending
- [3] Insert element at a position
- [4] Delete element from a position
- [5] Check if the list is empty
- [6] Display size

3

Enter the element to be inserted:

25

Enter the position:

3

Doubly linked list: 10 <-> 20 <-> 25 <-> 30 <-> 40 <-> 50

Press 'C' to continue and 'S' to stop

C

Select an operation:

- [1] Insert element at beginning
- [2] Insert element at ending
- [3] Insert element at a position
- [4] Delete element from a position
- [5] Check if the list is empty
- [6] Display size

4

Enter the position:

5

Doubly linked list: 10 <-> 20 <-> 25 <-> 30 <-> 50

Press 'C' to continue and 'S' to stop

C

Select an operation:

- [1] Insert element at beginning
- [2] Insert element at ending
- [3] Insert element at a position
- [4] Delete element from a position
- [5] Check if the list is empty
- [6] Display size

6

Size: 5

Doubly linked list: 10 <-> 20 <-> 25 <-> 30 <-> 50

Press 'C' to continue and 'S' to stop

S

<

Q.2] Write a program to implement circular linked list and its operations

```
/**
Name: KHAN MOHD OWAIIS RAZA
ID : 20BCD7138
Course: Data Structures & Algorithm
Code: CSE2001
Slot: L19+L20
**/
/* Lab-4 (01-10-2022)*/
/* Java code to implement circular linked list and its operations */
package CSE2001_Lab4_20BCD7138;
import java.util.Scanner;
class Node{
protected int data;
protected Node link;
public Node(){
link = null;
data = 0;
}
public Node(int d,Node n){
data = d;
link = n;
}
public void setLink(Node n){
link = n;
}
public void setData(int d){
data = d;
}
public Node getLink(){
return link;
}
public int getData(){
return data;
}}
class CLL{
protected Node start ;
protected Node end ;
```

```

public int size ;
public CLL(){
start = null;
end = null;
size = 0;
}
public boolean isEmpty(){
return start == null;
}
public int getSize(){
return size;
}
public void insertAtStart(int val){
Node nptr = new Node(val,null);
nptr.setLink(start);
if(start == null){
start = nptr;
nptr.setLink(start);
end = start;
}
else{
end.setLink(nptr);
start = nptr;
}
size++ ;
}
public void insertAtEnd(int val){
Node nptr = new Node(val,null);
nptr.setLink(start);
if(start == null){
start = nptr;
nptr.setLink(start);
end = start;
}
else{
end.setLink(nptr);
end = nptr;
}
size++ ;
}
public void insertAtPos(int val , int pos) {
Node nptr = new Node(val,null);
Node ptr = start;
pos = pos - 1 ;
for(int i = 1; i < size - 1; i++){
if (i == pos){
Node tmp = ptr.getLink() ;
ptr.setLink(nptr);
nptr.setLink(tmp);
break;
}
ptr = ptr.getLink();
}
size++ ;
}

```



```

}
public void deleteAtPos(int pos){
if (size == 1 && pos == 1){
start = null;
end = null;
size = 0;
return ;
}
if (pos == 1){
start = start.getLink();
end.setLink(start);
size--;
return ;
}
if (pos == size){
Node s = start;
Node t = start;
while (s != end){
t = s;
s = s.getLink();
}
end = t;
end.setLink(start);
size --;
return;
}
Node ptr = start;
pos = pos - 1 ;
for (int i = 1; i < size - 1; i++){
if (i == pos){
Node tmp = ptr.getLink();
tmp = tmp.getLink();
ptr.setLink(tmp);
break;
}
}
ptr = ptr.getLink();
}
size-- ;
}
public void display(){
System.out.print("\nCircular linked list: ");
Node ptr = start;
if (size == 0) {
System.out.print("Empty\n");
return;
}
if (start.getLink() == start){
System.out.print(start.getData()+ "->"+ptr.getData()+ "\n");
return;
}
System.out.print(start.getData()+ "->");
ptr = start.getLink();
while (ptr.getLink() != start){
System.out.print(ptr.getData()+ "->");

```

```

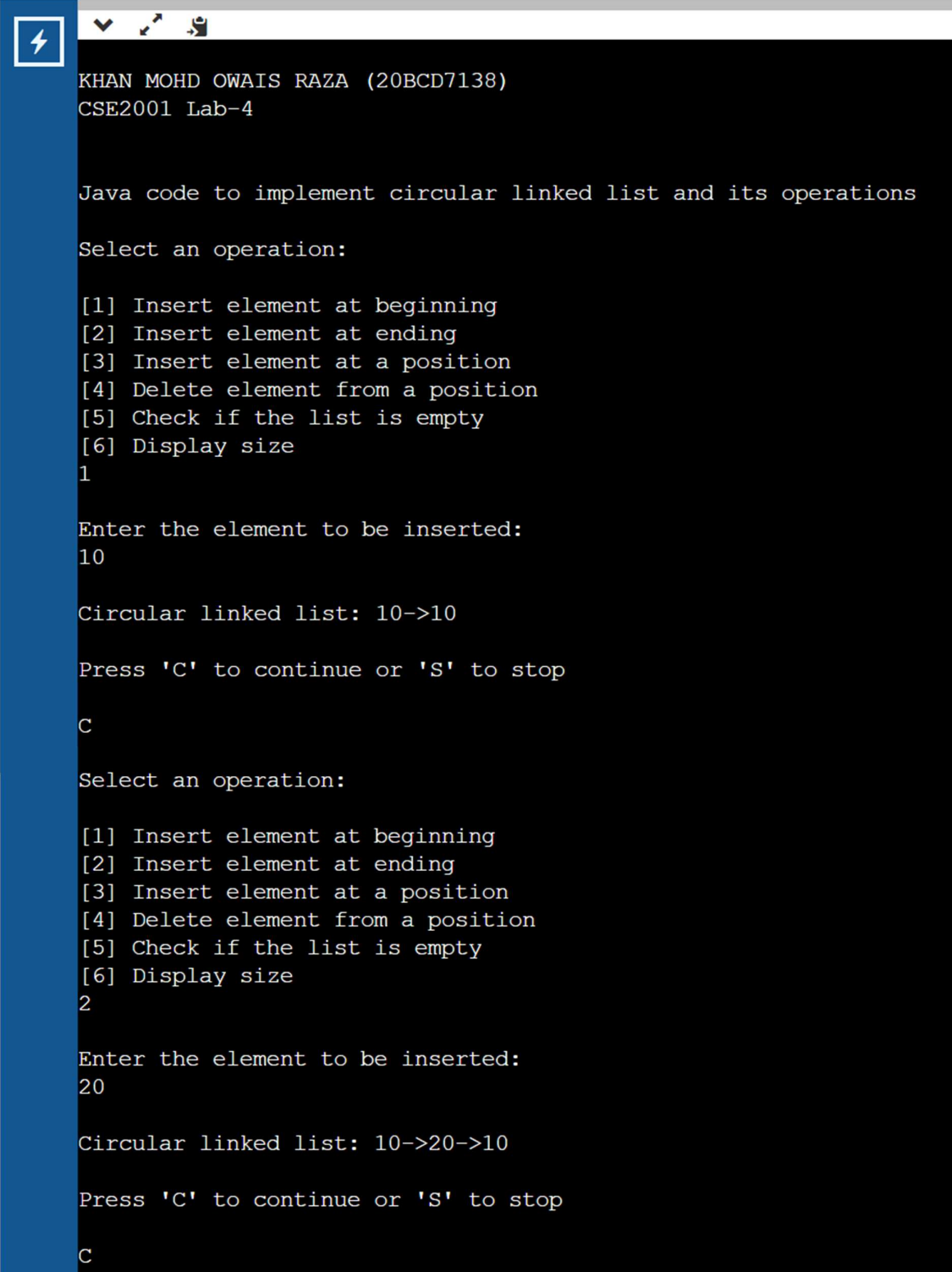
ptr = ptr.getLink();
}
System.out.print(ptr.getData()+ "->");
ptr = ptr.getLink();
System.out.print(ptr.getData()+ "\n");
}}
public class Question2{
public static void main(String[] args){
Scanner scan = new Scanner(System.in);
CLL list = new CLL();
System.out.print("\nKHAN MOHD OWAIS RAZA (20BCD7138)");
System.out.print("\nCSE2001 Lab-4");
System.out.print("\n");
System.out.print("\n");
System.out.print("\nJava code to implement circular linked list and its
operations");
System.out.print("\n");
char ch;
do{
System.out.println("\nSelect an operation: \n");
System.out.println("[1] Insert element at beginning");
System.out.println("[2] Insert element at ending");
System.out.println("[3] Insert element at a position");
System.out.println("[4] Delete element from a position");
System.out.println("[5] Check if the list is empty");
System.out.println("[6] Display size");
int choice = scan.nextInt();
switch (choice){
case 1 : System.out.println("\nEnter the element to be inserted: ");
list.insertAtStart( scan.nextInt() );
break;
case 2 :
System.out.println("\nEnter the element to be inserted: ");
list.insertAtEnd( scan.nextInt() );
break;
case 3 :
System.out.println("\nEnter the element to be inserted: ");
int num = scan.nextInt() ;
System.out.println("Enter position:");
int pos = scan.nextInt() ;
if (pos <= 1 || pos > list.getSize()) System.out.println("Invalid
position\n");
else list.insertAtPos(num, pos);
break;
case 4 : System.out.println("Enter position");
int p = scan.nextInt() ;
if (p < 1 || p > list.getSize()) System.out.println("Invalid position\n");
else list.deleteAtPos(p);
break;
case 5 : System.out.println("Empty status: " + list.isEmpty());
break;
case 6 :
System.out.println("Size = " + list.getSize() + " \n");
break;

```

```

default : System.out.println("Incorrect entry \n ");
break;
}
list.display();
System.out.println("\nPress 'C' to continue or 'S' to stop\n");
ch = scan.next().charAt(0);
} while (ch == 'C');
}}

```



```

KHAN MOHD OWAIS RAZA (20BCD7138)
CSE2001 Lab-4

Java code to implement circular linked list and its operations

Select an operation:

[1] Insert element at beginning
[2] Insert element at ending
[3] Insert element at a position
[4] Delete element from a position
[5] Check if the list is empty
[6] Display size
1

Enter the element to be inserted:
10

Circular linked list: 10->10

Press 'C' to continue or 'S' to stop
C

Select an operation:

[1] Insert element at beginning
[2] Insert element at ending
[3] Insert element at a position
[4] Delete element from a position
[5] Check if the list is empty
[6] Display size
2

Enter the element to be inserted:
20

Circular linked list: 10->20->10

Press 'C' to continue or 'S' to stop
C

```

Select an operation:

- [1] Insert element at beginning
- [2] Insert element at ending
- [3] Insert element at a position
- [4] Delete element from a position
- [5] Check if the list is empty
- [6] Display size

2

Enter the element to be inserted:

30

Circular linked list: 10->20->30->10

Press 'C' to continue or 'S' to stop

C

Select an operation:

- [1] Insert element at beginning
- [2] Insert element at ending
- [3] Insert element at a position
- [4] Delete element from a position
- [5] Check if the list is empty
- [6] Display size

2

Enter the element to be inserted:

40

Circular linked list: 10->20->30->40->10

Press 'C' to continue or 'S' to stop

C

Select an operation:

- [1] Insert element at beginning
- [2] Insert element at ending
- [3] Insert element at a position
- [4] Delete element from a position
- [5] Check if the list is empty
- [6] Display size

2

Enter the element to be inserted:

50

Circular linked list: 10->20->30->40->50->10

Press 'C' to continue or 'S' to stop

C

Select an operation:

- [1] Insert element at beginning
- [2] Insert element at ending
- [3] Insert element at a position
- [4] Delete element from a position
- [5] Check if the list is empty
- [6] Display size

3

Enter the element to be inserted:

25

Enter position:

3

Circular linked list: 10->20->25->30->40->50->10

Press 'C' to continue or 'S' to stop

C

Select an operation:

- [1] Insert element at beginning
- [2] Insert element at ending
- [3] Insert element at a position
- [4] Delete element from a position
- [5] Check if the list is empty
- [6] Display size

5

Empty status: false