**Practical 10**

**Name: Shantanu Sethi**

**Roll no.: 163**

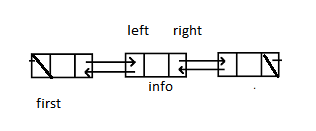
**Aim: To implement Doubly Link List**

**Objectives:**

1. Learn how to implement different operations on Doubly Link List
   * Insert from front
   * Insert from last
   * Delete from front
   * Delete from last
   * Display node in DLL
   * Count number of nodes in DLL
   * Search node in given DLL

**Theory:**

**Structure of DLL node :**

****

**Algorithm :**

**Insert from front in DLL:**

1. [Initialize field of new node]

info(Node1)= X

2. [ check Linked list is empty]

if(first==NULL)

first=Node1

left(Node1)=NULL

right(Node1)=NULL

Return first

3. [Insert in the beginning of DLL]

left(Node1)=NULL

right(Node1)=first

left(first)=Node1

first=Node1

**Insert node at the end of linked list**

1. [Initialize field of new node]

info(Node1)= X

2. [ check Linked list is empty]

if(first==NULL)

first=Node1

left(Node1)= NULL

right(Node1)=NULL

Return first

3. [Insert at the end of DLL]

Move=first

while(right(move) !=NULL)

move=right(move)

right(move)= Node1

left(Node1)=move

right(node1)= NULL

**Delete node from front of DLL**

1.[ check for empty DLL]

If(FIRST==NULL)

write (empty link list)

2.[Delete node if DLL containing single element]

if(RIGHT(FIRST)==NULL & LEFT(FIRST)==NULL)

write(INFO(FIRST))

3.[set first pointer to null]

FIRST=NULL

4.[delete first node ]

MOVE=RIGHT(FIRST)

write INFO(FIRST)

LEFT(MOVE)=NULL

RIGHT(FIRST)=NULL

FIRST=MOVE

**Delete node from last of DLL**

1.[ check for empty DLL]

If(FIRST==NULL)

write (empty link list)

2.[Delete node if DLL containing single element]

if(RIGHT(FIRST)==NULL & LEFT(FIRST)==NULL)

write(INFO(FIRST))

3.[set first pointer to null]

FIRST=NULL

4.[delete last node ]

MOVE=FIRST

while( RIGHT(MOVE)!=NULL)

PRED=MOVE

MOVE=RIGHT(MOVE)

write INFO(MOVE)

RIGHT(PRED)=NULL

**Print node from DLL**

1.[ check for empty DLL]

If(FIRST==NULL)

write (empty link list)

2..[print node ]

MOVE=FIRST

while( RIGHT(MOVE)!=NULL)

write(INFO(MOVE))

MOVE=RIGHT(MOVE)

**count node in DLL**

1.[ check for empty DLL]

If(FIRST==NULL)

write (empty link list)

C=0

2..[print node ]

MOVE=FIRST

while( RIGHT(MOVE)!=NULL)

C=C+1

MOVE=RIGHT(MOVE)

3. write( c )

**Program:**

package Doubly;

import java.util.Scanner;

public class Doubly {

public static void main(String[] args) {

// TODO Auto-generated method stub

System.out.println("Abhinav Singh-184");

Scanner scanner = new Scanner(System.in);

class Node {

int data;

Node prev, next;

Node(int data) {

this.data = data;

this.prev = null;

this.next = null;

}

}

Node first = null;

while (true) {

System.out.println("\n--- Doubly Linked List Operations ---");

System.out.println("1. Insert at front");

System.out.println("2. Insert at last");

System.out.println("3. Delete from front");

System.out.println("4. Delete from last");

System.out.println("5. Display nodes");

System.out.println("6. Count nodes");

System.out.println("7. Search for a node");

System.out.println("8. Exit");

System.out.print("Choose an option: ");

int choice = scanner.nextInt();

switch (choice) {

case 1:

System.out.print("Enter value to insert at front: ");

int valueFront = scanner.nextInt();

Node newNodeFront = new Node(valueFront);

if (first == null) {

first = newNodeFront;

} else {

newNodeFront.next = first;

first.prev = newNodeFront;

first = newNodeFront;

}

System.out.println("Inserted at front.");

break;

case 2:

System.out.print("Enter value to insert at last: ");

int valueLast = scanner.nextInt();

Node newNodeLast = new Node(valueLast);

if (first == null) {

first = newNodeLast;

} else {

Node temp = first;

while (temp.next != null) {

temp = temp.next;

}

temp.next = newNodeLast;

newNodeLast.prev = temp;

}

System.out.println("Inserted at last.");

break;

case 3:

if (first == null) {

System.out.println("Empty linked list.");

} else {

System.out.println("Deleted from front: " + first.data);

if (first.next == null) {

first = null;

} else {

first = first.next;

first.prev = null;

}

}

break;

case 4:

if (first == null) {

System.out.println("Empty linked list.");

} else {

Node temp = first;

while (temp.next != null) {

temp = temp.next;

}

System.out.println("Deleted from last: " + temp.data);

if (temp.prev == null) {

first = null;

} else {

temp.prev.next = null;

}

}

break;

case 5:

if (first == null) {

System.out.println("Empty linked list.");

} else {

Node temp = first;

System.out.print("Nodes in DLL: ");

while (temp != null) {

System.out.print(temp.data + " ");

temp = temp.next;

}

System.out.println();

}

break;

case 6:

int count = 0;

if (first == null) {

System.out.println("Empty linked list.");

} else {

Node temp = first;

while (temp != null) {

count++;

temp = temp.next;

}

System.out.println("Number of nodes: " + count);

}

break;

case 7:

System.out.print("Enter value to search: ");

int searchValue = scanner.nextInt();

boolean found = false;

if (first == null) {

System.out.println("Empty linked list.");

} else {

Node temp = first;

while (temp != null) {

if (temp.data == searchValue) {

found = true;

break;

}

temp = temp.next;

}

}

if (found) {

System.out.println("Node " + searchValue + " found in DLL.");

} else {

System.out.println("Node " + searchValue + " not found in DLL.");

}

break;

case 8:

System.out.println("Exiting...");

scanner.close();

return;

default:

System.out.println("Invalid option. Please try again.");

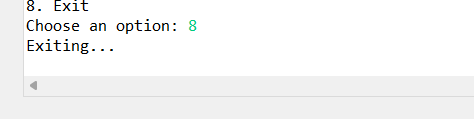
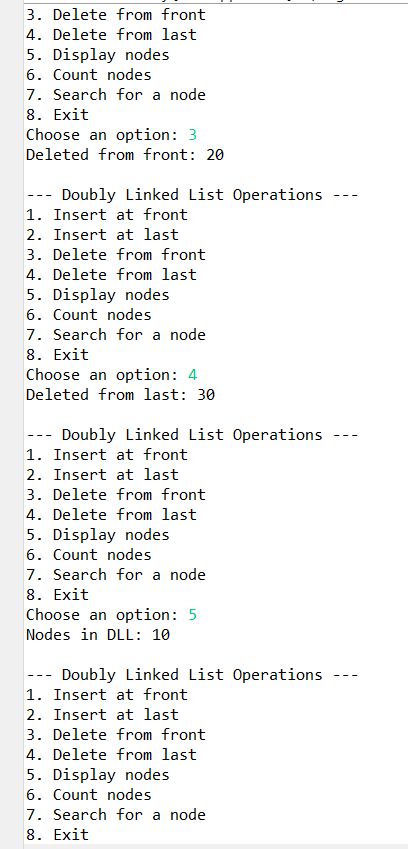
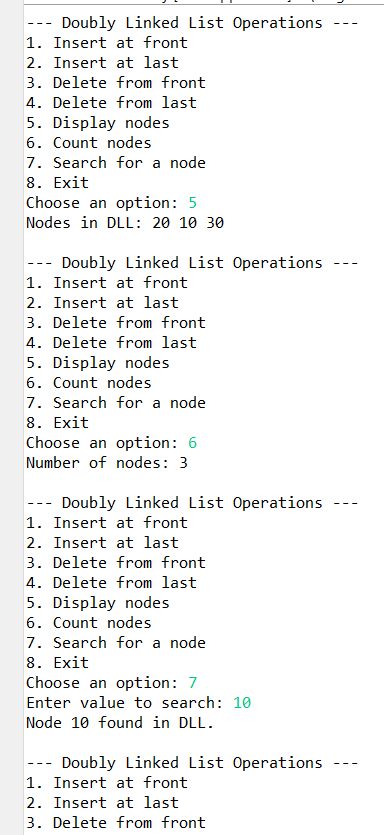
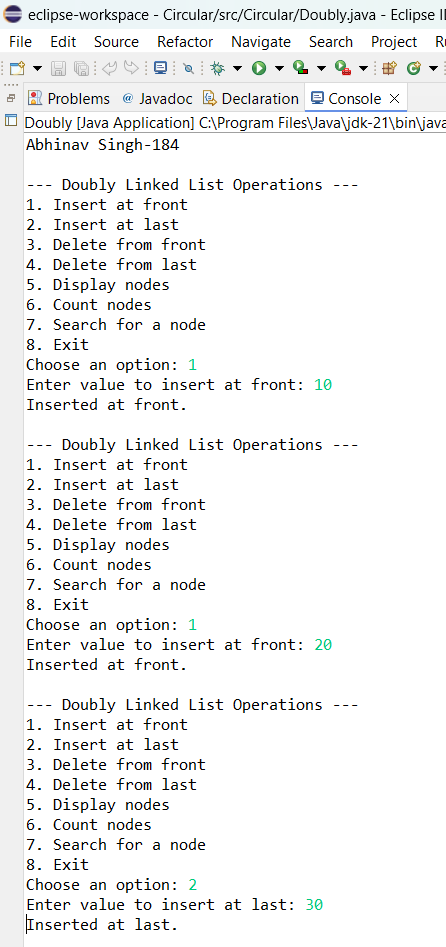
}

}

}

}

**Output:**

****

**Conclusion:** Successfully implemented different operations on DLL