**Topic – Implementing a Double Ended Queue Using Doubly Linked List**

* **Problem Statement**

Write a program to implement a double ended queue using doubly linked list. It should be a menu driven program with following options:  
  
1. Insert a new node at head  
2. Insert a new node at end  
3. Insert after kth node  
4. Delete from first  
5. Delete from end  
6. Delete the kth node  
7. Print the list  
8. Search an item in the list  
9. Exit

**Input and Output example:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

1

Enter the data to be inserted

10

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

1

Enter the data to be inserted

20

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

2

Enter the data to be inserted

30

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

2

Enter the data to be inserted

40

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

7

The linked list is :20 10 30 40

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

3

Enter the data to be inserted

5

Enter the position of the node after which a new node is to be inserted

0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

7

The linked list is :5 20 10 30 40

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

3

Enter the data to be inserted

35

Enter the position of the node after which a new node is to be inserted

4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

7

The linked list is :5 20 10 30 35 40

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

3

Enter the data to be inserted

7

Enter the position of the node after which a new node is to be inserted

7

Invalid position is given

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

7

The linked list is :5 20 10 30 35 40

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

7

The linked list is :20 10 30 35 40

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

5

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

7

The linked list is :20 10 30 35

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

6

Enter the position of the node to be deleted

2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

7

The linked list is :20 30 35

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

6

Enter the position of the node to be deleted

4

Invalid position is given

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

7

The linked list is :20 30 35

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

8

Enter the data to be searched

30

30 is present in the position 2 of the linked list

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

8

Enter the data to be searched

18

18 is not present in the linked list

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

5

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

5

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

7

No linked list is present

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

5

Deletion is not required as no linked list is present

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

8

Enter the data to be searched

7

No linked list is present

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

1

Enter the data to be inserted

10

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

7

The linked list is :10

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1:Insert a new node at head

2:Insert a new node at end

3:Insert after kth node

4:Delete from first

5:Delete from end

6:Delete the kth node

7:Print the list

8:Search an item in the list

9:exit

\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*

9

/\* Here the program terminates \*/

* **Proposed C Code**

**/\* ---------- doublylinkedlist.c--------------- \*/**

**#include<stdio.h>**

**#include<stdlib.h>**

**/\* A self referential structure is declared as name Node using typedef keward \*/**

**/\* It contains data and pointer previous and next of same type Node to define a doubly linked list \*/**

**typedef struct node**

**{**

**int data;**

**struct node \* previous;**

**struct node \* next;**

**}Node;**

**/\* head and tail are the ceninal nodes (like front and rear) globally declared and initially set to NULL \*/**

**Node\* head = NULL;**

**Node\* tail = NULL;**

**/\* A function to create a node \*/**

**Node \* create(int data)**

**{**

**Node\* create = (Node\*)malloc(sizeof(Node));**

**create->data= data;**

**create->previous = NULL;**

**create->next = NULL;**

**return create;**

**}**

**/\* A function to find the length of the linked list \*/**

**int linkedlistSize()**

**{**

**Node\* temp = head;**

**int length = 0;**

**while ( temp != NULL)**

**{**

**temp = temp->next;**

**length++;**

**}**

**temp = NULL;**

**return length;**

**}**

**/\* A function to insert a node at the first of the doubly linked list(Enqueueing at first) \*/**

**Node\*insertFirst(int data)**

**{**

**Node\* temp = create(data);/\* create(data) is called to insert the required node \*/**

**if ( head == NULL )**

**{**

**head = temp;/\* head == NULL means no linked list so temp is new head and tail is also pointed to it \*/**

**tail = temp;**

**}**

**else**

**{**

**/\* inserting temp at first position by joining temp->next to previous head and its previous to temp \*/**

**temp->next = head;**

**head->previous = temp;**

**head = temp;**

**/\* as temp is at the begining so temp is the new head \*/**

**}**

**temp = NULL;**

**/\* as we got our head so temp is not required and it is nullified to avoid garbage value \*/**

**return head;**

**}**

**/\* function to insert node at end of the doubly linked list(Enqueueing at end) \*/**

**Node\*insertEnd(int data)**

**{**

**Node\* temp = create(data); /\* creating node \*/**

**if ( head == NULL )**

**{**

**head = temp;/\* if no linked list then temp is new head and tail is also pointed to it \*/**

**tail = temp;**

**}**

**else**

**{**

**/\* as temp is at last,so its previous is joined to tail and its next is set to NULL when the create(data) was called \*/**

**temp->previous = tail;**

**tail->next = temp; /\* temp is joined after the last node pointed by tail \*/**

**tail = temp; /\* temp is the new tail \*/**

**}**

**/\* as temp is not required more so it is nullified to avoid garbage value \*/**

**temp = NULL;**

**return head;**

**}**

**/\* inserting new node after kth node where k is the position of that node \*/**

**Node\*insertPosition(int data,int position)**

**{**

**Node\* temp = create(data); /\* creating node \*/**

**int length = linkedlistSize(); /\* the length of the linked list is calculated \*/**

**/\* if no linked list is there so insertion can take place only at 0th index \*/**

**if (( length == 0) && ( position != 0))**

**{**

**printf("\n No node is present,so insertion will be possible in 0th index\n");**

**return head;**

**}**

**/\* insertion is possible upto the after the last node i.e upto the position length \*/**

**else if (( position < 0 ) || ( position > length ))**

**{**

**printf("\n Invalid position is given\n");**

**return head;**

**}**

**else**

**{**

**if ( position == 0 ) /\* 0th index means inserting before first node or at very first position \*/**

**{**

**head = insertFirst(data);**

**return head;**

**}**

**else if ( position == length ) /\* inserting after the last node \*/**

**{**

**head = insertEnd(data);**

**return head;**

**}**

**else**

**{**

**Node\* ptr = head;**

**for ( int i = 0 ; i < position-1 ; i++ )**

**{**

**ptr = ptr->next;**

**}**

**/\* ptr is set to the node after which temp is to be inserted \*/**

**/\* the connections are done at the required position \*/**

**temp->previous = ptr;**

**temp->next = ptr->next;**

**ptr->next->previous = temp;**

**ptr->next = temp;**

**/\* as temp,ptr are not required more so they are nullified to avoid garbage value \*/**

**ptr = NULL;**

**temp = NULL;**

**return head;**

**}**

**}**

**}**

**/\* deleting node from the first(dequeueing from first) \*/**

**Node\* deleteFirst()**

**{**

**if ( head == NULL )**

**{**

**printf("\n Deletion is not required as no linked list is present\n");**

**}**

**else**

**{**

**Node\* ptr = head;**

**/\* updating the head to head->next \*/**

**head = head->next;**

**if ( head != NULL )**

**{**

**head->previous = NULL; /\* the new head's previous is set to NULL \*/**

**}**

**else**

**{**

**tail = NULL; /\* new head is NULL means no node is there so tail is also updated to NULL \*/**

**}**

**/\* deleting previous head now pointed by ptr \*/**

**free(ptr); /\* freeing the first node \*/**

**}**

**return head;**

**}**

**/\* deleting node at end(dequeueing from end) \*/**

**Node\* deleteEnd()**

**{**

**if ( head == NULL )**

**{**

**printf("\n Deletion is not required as no linked list is present\n");**

**}**

**else**

**{**

**Node\* ptr = tail;**

**if ( tail->previous != NULL )**

**{**

**/\* as tail->previous will be the new tail so its next is set to NULL and it is set as new tail \*/**

**tail->previous->next = NULL;**

**tail = tail->previous;**

**}**

**else**

**{**

**/\* tail-> pprevious is NULL means only one node is there which is to be deleted so head,tail is set to NULL \*/**

**tail = NULL;**

**head = NULL;**

**}**

**free(ptr); /\* freeing the last node \*/**

**}**

**return head;**

**}**

**/\* deleting kth node where k is the position of that node \*/**

**Node\* deletePosition(int position)**

**{**

**int length = linkedlistSize(); /\* the length of the linked list is calculated \*/**

**if ( length == 0)**

**{**

**printf("\n Deletion is not required as no node is present\n");**

**return head;**

**}**

**else if (( position < 1 ) || ( position > length )) /\* condition for invalid position \*/**

**{**

**printf("\n Invalid position is given\n");**

**return head;**

**}**

**else**

**{**

**if ( position == 1 ) /\*deleting the first node \*/**

**{**

**head = deleteFirst();**

**return head;**

**}**

**else if ( position == length ) /\*deleting the last node \*/**

**{**

**head = deleteEnd();**

**return head;**

**}**

**else**

**{**

**Node\* ptr = head;**

**for ( int i = 1 ; i < position ; i++ )**

**{**

**ptr = ptr->next;**

**}**

**/\* ptr is set to the node to be deleted \*/**

**/\* the connections are done between the previous and next node of the node epoined by ptr \*/**

**ptr->previous->next =ptr->next;**

**ptr->next->previous = ptr->previous;**

**free(ptr); /\* freeing the node \*/**

**return head;**

**}**

**}**

**}**

**/\* function to print the doubly linked list \*/**

**void print(Node\* head)**

**{**

**if(head==NULL)**

**{**

**printf("\n No linked list is present\n");**

**}**

**else**

**{**

**Node\* ptr= head;**

**printf("\n The linked list is :");**

**while(ptr!=NULL) /\*Traversing the linked list \*/**

**{**

**printf("%d ",ptr->data);**

**ptr=ptr->next;**

**}**

**printf("\n");**

**}**

**}**

**/\* function to search item in the list \*/**

**void search(int data)**

**{**

**if ( head == NULL )**

**{**

**printf("\n No linked list is present\n");**

**}**

**else**

**{**

**Node\* ptr=head;**

**int i = 1; /\* i is required to identify position where the required data is present \*/**

**while ( ptr != NULL )/\* As ptr reaches null means it crosses the linked list \*/**

**{**

**if( ptr->data == data )/\* Searching the position where data is present \*/**

**{**

**break;**

**}**

**i++;**

**ptr=ptr->next;**

**}**

**if(ptr==NULL) /\*ptr is NULL means required data is not present in the entire linked list after traversal \*/**

**{**

**printf("\n %d is not present in the linked list\n",data);**

**}**

**else**

**{**

**printf("\n %d is present in the position %d of the linked list\n",data,i);**

**/\* printing the position of the searched data in the linked list \*/**

**}**

**}**

**}**

**int main()**

**{**

**int option,data,position;**

**while(1){**

**/\* the options are given \*/**

**printf("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Menu of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");**

**printf("1:Insert a new node at head\n");**

**printf("2:Insert a new node at end\n");**

**printf("3:Insert after kth node\n");**

**printf("4:Delete from first\n");**

**printf("5:Delete from end\n");**

**printf("6:Delete the kth node\n");**

**printf("7:Print the list\n");**

**printf("8:Search an item in the list\n");**

**printf("9:exit\n");**

**printf("\n\*\*\*\*\*\*\*\*Enter option\*\*\*\*\*\*\*\*\n");**

**scanf("%d",&option);**

**switch(option)/\* the cases are designed according to the opertions listed above \*/**

**{**

**case 1:**

**printf("Enter the data to be inserted\n");**

**scanf("%d",&data);**

**head= insertFirst(data);**

**break;**

**case 2:**

**printf("Enter the data to be inserted\n");**

**scanf("%d",&data);**

**head= insertEnd(data);**

**break;**

**case 3:**

**printf("Enter the data to be inserted\n");**

**scanf("%d",&data);**

**printf("Enter the position of the node after which a new node is to be inserted\n");**

**scanf("%d",&position);**

**head= insertPosition(data,position);**

**break;**

**case 4:**

**head= deleteFirst();**

**break;**

**case 5:**

**head= deleteEnd();**

**break;**

**case 6:**

**printf("Enter the position of the node to be deleted\n");**

**scanf("%d",&position);**

**head= deletePosition(position);**

**break;**

**case 7:**

**print(head);**

**break;**

**case 8:**

**printf("Enter the data to be searched\n");**

**scanf("%d",&data);**

**search(data);**

**break;**

**case 9:**

**break; /\* to exit from the loop if option is 9 is mentioned after default in the switch case \*/**

**default:**

**printf("Please give option withen (1-9)\n");**

**}**

**/\* exit from the loop after option 9 is chosen \*/**

**if ( option == 9)**

**{**

**break;**

**}**

**}**

**int length = linkedlistSize(); /\* finding the length of the linked list for deletion \*/**

**/\* as all the operations are done so the memory allocated by the linked list is freed \*/**

**for( int i = 0 ; i < length ; i++)**

**{**

**head = deleteEnd();**

**}**

**return 0;**

**}**

**/\*------------------------------------------------------------------------------------------------------------------------- \*/**

* **Conclusion**

**The proposed algorithm has overall runtime of O(n) where n is number of opeartions given by the user as options before exit from the program. Also for the specific functions like printing the list or deleteing or inserting at given position takes time of O(n) where n is the length of the linked list,for inserting at first or end or deleting from first or end takes O(1) as we used head and tail for enqueueing and dequeueing from first or end .**

* **Limitations : Initially the doubly linked list is taken as empty. We should adjust the size of the linked list in such a way that it could accommodate withen the heap segment of the memory.**
* **Assumptions: Initially the doubly linked list is taken as empty.**