**1. Topic – Different Operations on Linked List**

* **Problem Statement**

**Write C Program to show the diffrent operations on a singly linked list. Let the options 0-10 indicates the following operations:**

**0: Exit  
1: Insert a node  in the begining of the linked list  
2. Insert a node  at the end of the linked list  
3. Insert a node  at the kth position of the linked list  
4. Search a node containing a given value  
5. Print the entire linked list  
6. Delete a node from the begining of the linked list  
7. Dele a node  at the end of the linked list  
8.Delete  a node  at the kth position of the linked list  
9. Continue  
   
The user should be able to give his/her option to perform different operations on a singly linked list using the above menu as long as he/ she wants.**

**Input and Output example:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*List of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0:Exit

1:Insert a node in the begining of the linked list

2:Insert a node at the end of the linked list

3:Insert a node at the kth position of the linked list

4:Search a node containing a given value

5:Print the entire linked list

6:Delete a node from the begining of the linked list

7:Delete a node at the end of the linked list

8:Delete a node at the kth position of the linked list

9:continue

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

1

Enter the data to be inserted

7

\*\*\*\*\*\*\*\*\*\*\*\*\*\*List of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0:Exit

1:Insert a node in the begining of the linked list

2:Insert a node at the end of the linked list

3:Insert a node at the kth position of the linked list

4:Search a node containing a given value

5:Print the entire linked list

6:Delete a node from the begining of the linked list

7:Delete a node at the end of the linked list

8:Delete a node at the kth position of the linked list

9:continue

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

1

Enter the data to be inserted

9

\*\*\*\*\*\*\*\*\*\*\*\*\*\*List of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0:Exit

1:Insert a node in the begining of the linked list

2:Insert a node at the end of the linked list

3:Insert a node at the kth position of the linked list

4:Search a node containing a given value

5:Print the entire linked list

6:Delete a node from the begining of the linked list

7:Delete a node at the end of the linked list

8:Delete a node at the kth position of the linked list

9:continue

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

5

The linked list is :9 7

\*\*\*\*\*\*\*\*\*\*\*\*\*\*List of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0:Exit

1:Insert a node in the begining of the linked list

2:Insert a node at the end of the linked list

3:Insert a node at the kth position of the linked list

4:Search a node containing a given value

5:Print the entire linked list

6:Delete a node from the begining of the linked list

7:Delete a node at the end of the linked list

8:Delete a node at the kth position of the linked list

9:continue

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

2

Enter the data to be inserted

6

\*\*\*\*\*\*\*\*\*\*\*\*\*\*List of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0:Exit

1:Insert a node in the begining of the linked list

2:Insert a node at the end of the linked list

3:Insert a node at the kth position of the linked list

4:Search a node containing a given value

5:Print the entire linked list

6:Delete a node from the begining of the linked list

7:Delete a node at the end of the linked list

8:Delete a node at the kth position of the linked list

9:continue

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

5

The linked list is :9 7 6

\*\*\*\*\*\*\*\*\*\*\*\*\*\*List of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0:Exit

1:Insert a node in the begining of the linked list

2:Insert a node at the end of the linked list

3:Insert a node at the kth position of the linked list

4:Search a node containing a given value

5:Print the entire linked list

6:Delete a node from the begining of the linked list

7:Delete a node at the end of the linked list

8:Delete a node at the kth position of the linked list

9:continue

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

3

Enter the data to be inserted

4

Enter the position where to be inserted

4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*List of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0:Exit

1:Insert a node in the begining of the linked list

2:Insert a node at the end of the linked list

3:Insert a node at the kth position of the linked list

4:Search a node containing a given value

5:Print the entire linked list

6:Delete a node from the begining of the linked list

7:Delete a node at the end of the linked list

8:Delete a node at the kth position of the linked list

9:continue

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

5

The linked list is :9 7 6 4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*List of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0:Exit

1:Insert a node in the begining of the linked list

2:Insert a node at the end of the linked list

3:Insert a node at the kth position of the linked list

4:Search a node containing a given value

5:Print the entire linked list

6:Delete a node from the begining of the linked list

7:Delete a node at the end of the linked list

8:Delete a node at the kth position of the linked list

9:continue

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

2

Enter the data to be inserted

1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*List of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0:Exit

1:Insert a node in the begining of the linked list

2:Insert a node at the end of the linked list

3:Insert a node at the kth position of the linked list

4:Search a node containing a given value

5:Print the entire linked list

6:Delete a node from the begining of the linked list

7:Delete a node at the end of the linked list

8:Delete a node at the kth position of the linked list

9:continue

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

3

Enter the data to be inserted

12

Enter the position where to be inserted

2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*List of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0:Exit

1:Insert a node in the begining of the linked list

2:Insert a node at the end of the linked list

3:Insert a node at the kth position of the linked list

4:Search a node containing a given value

5:Print the entire linked list

6:Delete a node from the begining of the linked list

7:Delete a node at the end of the linked list

8:Delete a node at the kth position of the linked list

9:continue

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

5

The linked list is :9 12 7 6 4 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*List of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0:Exit

1:Insert a node in the begining of the linked list

2:Insert a node at the end of the linked list

3:Insert a node at the kth position of the linked list

4:Search a node containing a given value

5:Print the entire linked list

6:Delete a node from the begining of the linked list

7:Delete a node at the end of the linked list

8:Delete a node at the kth position of the linked list

9:continue

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

4

Enter the data to be searched

6

6 is present in the position 4 of the linked list

\*\*\*\*\*\*\*\*\*\*\*\*\*\*List of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0:Exit

1:Insert a node in the begining of the linked list

2:Insert a node at the end of the linked list

3:Insert a node at the kth position of the linked list

4:Search a node containing a given value

5:Print the entire linked list

6:Delete a node from the begining of the linked list

7:Delete a node at the end of the linked list

8:Delete a node at the kth position of the linked list

9:continue

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

4

Enter the data to be searched

56

56 is not present in the linked list

\*\*\*\*\*\*\*\*\*\*\*\*\*\*List of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0:Exit

1:Insert a node in the begining of the linked list

2:Insert a node at the end of the linked list

3:Insert a node at the kth position of the linked list

4:Search a node containing a given value

5:Print the entire linked list

6:Delete a node from the begining of the linked list

7:Delete a node at the end of the linked list

8:Delete a node at the kth position of the linked list

9:continue

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

5

The linked list is :9 12 7 6 4 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*List of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0:Exit

1:Insert a node in the begining of the linked list

2:Insert a node at the end of the linked list

3:Insert a node at the kth position of the linked list

4:Search a node containing a given value

5:Print the entire linked list

6:Delete a node from the begining of the linked list

7:Delete a node at the end of the linked list

8:Delete a node at the kth position of the linked list

9:continue

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

6

\*\*\*\*\*\*\*\*\*\*\*\*\*\*List of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0:Exit

1:Insert a node in the begining of the linked list

2:Insert a node at the end of the linked list

3:Insert a node at the kth position of the linked list

4:Search a node containing a given value

5:Print the entire linked list

6:Delete a node from the begining of the linked list

7:Delete a node at the end of the linked list

8:Delete a node at the kth position of the linked list

9:continue

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

7

\*\*\*\*\*\*\*\*\*\*\*\*\*\*List of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0:Exit

1:Insert a node in the begining of the linked list

2:Insert a node at the end of the linked list

3:Insert a node at the kth position of the linked list

4:Search a node containing a given value

5:Print the entire linked list

6:Delete a node from the begining of the linked list

7:Delete a node at the end of the linked list

8:Delete a node at the kth position of the linked list

9:continue

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

5

The linked list is :12 7 6 4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*List of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0:Exit

1:Insert a node in the begining of the linked list

2:Insert a node at the end of the linked list

3:Insert a node at the kth position of the linked list

4:Search a node containing a given value

5:Print the entire linked list

6:Delete a node from the begining of the linked list

7:Delete a node at the end of the linked list

8:Delete a node at the kth position of the linked list

9:continue

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

8

Enter the position

2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*List of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0:Exit

1:Insert a node in the begining of the linked list

2:Insert a node at the end of the linked list

3:Insert a node at the kth position of the linked list

4:Search a node containing a given value

5:Print the entire linked list

6:Delete a node from the begining of the linked list

7:Delete a node at the end of the linked list

8:Delete a node at the kth position of the linked list

9:continue

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

5

The linked list is :12 6 4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*List of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0:Exit

1:Insert a node in the begining of the linked list

2:Insert a node at the end of the linked list

3:Insert a node at the kth position of the linked list

4:Search a node containing a given value

5:Print the entire linked list

6:Delete a node from the begining of the linked list

7:Delete a node at the end of the linked list

8:Delete a node at the kth position of the linked list

9:continue

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

9

\*\*\*\*\*\*\*\*\*\*\*\*\*\*List of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0:Exit

1:Insert a node in the begining of the linked list

2:Insert a node at the end of the linked list

3:Insert a node at the kth position of the linked list

4:Search a node containing a given value

5:Print the entire linked list

6:Delete a node from the begining of the linked list

7:Delete a node at the end of the linked list

8:Delete a node at the kth position of the linked list

9:continue

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

11

Please give option withen (0-9)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*List of options\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0:Exit

1:Insert a node in the begining of the linked list

2:Insert a node at the end of the linked list

3:Insert a node at the kth position of the linked list

4:Search a node containing a given value

5:Print the entire linked list

6:Delete a node from the begining of the linked list

7:Delete a node at the end of the linked list

8:Delete a node at the kth position of the linked list

9:continue

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

0

/\* Here the program terminates \*/

* **Proposed C Code**

**/\* ---------- linkedlist.c--------------- \*/**

#include<stdio.h>

#include<stdlib.h>

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

/\* It contains data and ponter next of same type of Node\*/

typedef struct node

{

int data;

struct node \* next;

}Node;

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

Node \* create(int data)

{

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

create->data= data;

create->next = NULL;

return create;

}

/\* A function to insert a node to the begining of a linked list \*/

Node\*insertBegin(Node\* head,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 \*/

}

else

{

/\* inserting temp at begining by first joining temp->next to previous head\*/

temp->next=head;

head = temp;

/\* as temp is at 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 \*/

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

{

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

if(head==NULL)

{

head = temp;/\* if no linked list then temp is new head \*/

}

else

{

Node\* ptr= head;

/\* Traversing to the last of the linked list before NULL \*/

while (ptr->next != NULL)

{

ptr=ptr->next;

}

ptr->next=temp; /\* temp is joined after the last node pointer \*/

ptr=NULL;

/\* as ptr is not quired and it is nullified to avoid garbage value\*/

}

temp=NULL;

return head;

}

/\* inserting node at a given position \*/

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

{

Node\* temp = create(data);

if ((head==NULL) && (position==1))

{

head = temp;/\* if no linked list is present then insertion is possible in the 1st position only \*/

return head;

}

else

{

int i=1;/\* starting variable for iteration as position starts from 1 so i is initialized to 1 \*/

Node\* ptr=head;

Node\* previous=NULL; /\* previous pointer is required which follows ptr \*/

while (i < position)

{

previous = ptr;

if(ptr==NULL)

{

/\* If previous enters the loop and then becomes NULL\*/

/\* ptr is null it means it reached the end but previos is null means it crossed the length of linked list \*/

/\* So position is invalid so head is returned from the function \*/

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

return head;

}

ptr = ptr->next;

/\* ptr is updated upto position and previous pointer is updated to the previous value of ptr \*/

i++;/\* updating i for iteration \*/

}

if(position <= 0)

{

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

return head;

}

/\* previous is NULL but it did not enter the loop so position >0 and must be 1 as it starts with 1\*/

/\*otherwise it would return from the function in while loop if previous becomes null after crossing the linked list \*/

if((previous == NULL) && (head != NULL))

{

/\* if position is 1 and head is not null inserting node after head \*/

temp->next=head;

head=temp;

}

else

{

/\* inserting node pointed by temp by joining to ptr as ptr is in the position where temp is to be joined \*/

temp->next=ptr;

/\* previous of ptr is now joind to temp to make new connection \*/

previous->next=temp;

}

temp=NULL;

previous=NULL;

ptr=NULL;

/\* All used variables are nullified as desired linked list is formed \*/

return head;

}

}

/\* function to search a node with given value \*/

void search(Node\* head,int data)

{

Node\* ptr=head;

if(head==NULL)

{

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

}

else

{

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

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

{

ptr=ptr->next;

i++;

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

{

break;

}

}

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 \*/

}

}

}

/\* deleting node at the begining \*/

Node\* deleteBegin(Node\* head)

{

if(head==NULL)

{

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

}

else

{

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

Node\* ptr= head;

head= head->next;

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

free(ptr);

}

return head;

}

/\* deleting node at end \*/

Node\* deleteEnd(Node\* head)

{

if(head==NULL)

{

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

}

else

{

Node\* ptr= head;

Node\* previous= NULL;

while(ptr->next!=NULL)/\* updating ptr to last node and "previous" pointer previous to the last node \*/

{

previous= ptr;

ptr=ptr->next;

}

if(previous == NULL)

{

head = head->next;

}

else

{

previous->next=NULL;

/\* previous points to the last node after deletion so previous->next = NULL \*/

}

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

}

return head;

}

/\* deleting node from a given position \*/

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

{

if(head==NULL)

{

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

return head;

}

else

{

Node\* previous= NULL;/\* previous pointer is required which follows ptr \*/

Node\* ptr=head;

int i= 1;/\* starting variable for iteration as position starts from 1 so i is initialized to 1 \*/

while ( i < position )

{

/\* ptr is updated upto position and previous pointer is updated to the previous value of ptr \*/

previous= ptr;

ptr=ptr->next;

if(ptr==NULL)/\* ptr reaches null means it has crossed the entire linked list \*/

{

printf("\n Invalid position is given by the user\n");

return head;

}

i++;/\* updating i for iteration \*/

}

if(position <= 0)

{

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

return head;

}

/\* previous is NULL but it did not enter the loop so position >0 and must be 1 as it starts with 1\*/

/\*otherwise it would return from the function in while loop if previous becomes null after crossing the linked list \*/

if(previous==NULL)

{

/\* as for this condition position is 1 so head is updated to head->next and ptr is previous head then\*/

head = head->next;

}

else

{

/\* previous->next is joined to ptr->next as ptr is in the position where deletion is required \*/

previous->next = ptr->next;

}

free(ptr);/\* deleting the required node at the given position pointed by ptr \*/

return head;

}

}

/\* function to print the 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");

}

}

int main()

{

int option,data,position;

Node\* head= NULL;/\* initially no linked list is given \*/

do{

/\* the options are given \*/

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

printf("0:Exit\n");

printf("1:Insert a node in the begining of the linked list\n");

printf("2:Insert a node at the end of the linked list\n");

printf("3:Insert a node at the kth position of the linked list\n");

printf("4:Search a node containing a given value\n");

printf("5:Print the entire linked list\n");

printf("6:Delete a node from the begining of the linked list\n");

printf("7:Delete a node at the end of the linked list\n");

printf("8:Delete a node at the kth position of the linked list\n");

printf("9:continue\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= insertBegin(head,data);

break;

case 2:

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

scanf("%d",&data);

head= insertEnd(head,data);

break;

case 3:

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

scanf("%d",&data);

printf("Enter the position where to be inserted\n");

scanf("%d",&position);

head= insertPosition(head,data,position);

break;

case 4:

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

scanf("%d",&data);

search(head,data);

break;

case 5:

print(head);

break;

case 6:

head= deleteBegin(head);

break;

case 7:

head= deleteEnd(head);

break;

case 8:

printf("Enter the position\n");

scanf("%d",&position);

head= deletePosition(head,position);

break;

case 9:

continue;

break;

default:

if(option == 0)

{

exit(0);

}

else

{

printf("Please give option withen (0-9)\n");

}

}

}while(option!=0); /\* when option is zero then it will exit \*/

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.**

* **Limitations : Initially the 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 linked list is taken as empty.**