| **Ex. No.:01** | **Implementation of Single Linked List** | **Date:29/02/2024** |
| --- | --- | --- |

**Write a C program to implement the following operations on Singly Linked List.**

1. **Insert a node in the beginning of a list.**
2. **Insert a node after P**
3. **Insert a node at the end of a list**
4. **Find an element in a list**
5. **FindNext**
6. **FindPrevious**
7. **isLast**
8. **isEmpty**
9. **Delete a node in the beginning of a list.**
10. **Delete a node after P**
11. **Delete a node at the end of a list**
12. **Delete the List**

**Algorithm:**

1. Start
2. Define a Node structure with data and a pointer to the next Node.
3. Initialize a head pointer to NULL.
4. Create functions for the following operations:
5. Insert at the beginning or end:

* Create a new Node with the given data.
* Traverse the list to the last Node or update the head pointer accordingly.

1. Delete from the beginning or end:

* Update the head pointer to the next Node or traverse to the second last Node and update its pointer.

1. Search for a value:

* Traverse the list and compare each Node's data with the given value.
* Return the Node if found, otherwise return NULL.

1. Test the operations by inserting, deleting, and searching for elements in the list.
2. Stop

**PROGRAM:**

**#include <stdio.h> #include<stdlib.h>**

**void createfnode(int ele); void insertfront(int ele); void insertend(int ele); void display();**

**//type declaration of a node struct node**

**{**

**int data;**

**struct node\* next;**

**};**

**struct node\* head = NULL; struct node \*newnode; void insertfront(int ele)**

**{**

**newnode=(struct node\*)malloc(sizeof(struct node)); if(newnode!=NULL)**

**{ newnode->data=ele; if(head!=NULL)**

**{**

**newnode->next=head; head=newnode;**

**}**

**else**

**{**

**newnode->next=NULL; head=newnode;**

**}**

**}**

**}**

**void insertend(int ele)**

**{**

**newnode=(struct node\*)malloc(sizeof(struct node)); if(newnode!=NULL)**

**{**

**newnode->data=ele; newnode->next=NULL; if(head!=NULL)**

**{**

**struct node \*t; t=head;**

**while(t->next!=NULL)**

**{**

**t=t->next;**

**}**

**newnode->next=NULL; t->next=newnode;**

**}**

**else**

**{**

**head=newnode;**

**}**

**}**

**}**

**int listsize()**

**{**

**int c=0; struct node \*t; t=head; while(t!=NULL)**

**{**

**c=c+1; t=t->next;**

**}**

**printf("\n The size of the list is %d:\n",c); return c;**

**}**

**void insertpos(int ele,int pos)**

**{**

**int ls=0; ls=listsize();**

**if(head == NULL && (pos <= 0 || pos > 1))**

**{**

**printf("\nInvalid position to insert a node\n"); return;**

**}**

**// if the list is not empty and the position is out of range if(head != NULL && (pos <= 0 || pos > ls))**

**{**

**printf("\nInvalid position to insert a node\n"); return;**

**}**

**struct node\* newnode = NULL;**

**newnode=(struct node\*)malloc(sizeof(struct node)); if(newnode != NULL)**

**{**

**newnode->data=ele; struct node\* temp = head;**

**//getting the position-1 node int count = 1;**

**while(count < pos-1)**

**{**

**temp = temp -> next; count += 1;**

**}**

**//if the position is 1 then insertion at the beginning if(pos == 1)**

**{**

**newnode->next = head; head = newnode;**

**}**

**else**

**{**

**newnode->next = temp->next; temp->next = newnode;**

**}**

**}**

**}**

**void findnext(int s)**

**{**

**struct node \*temp; temp=head;**

**if(temp==NULL&&temp->next==NULL)**

**{**

**printf("No next element ");**

**}**

**else**

**{**

**while(temp->data!=s)**

**{**

**temp=temp->next;**

**}**

**printf("\nNext Element of %d is %d\n",s,temp->next->data);**

**}**

**}**

**void findprev(int s)**

**{**

**struct node \*temp; temp=head; if(temp==NULL)**

**{**

**printf("List is empty ");**

**}**

**else**

**{**

**while(temp->next->data!=s)**

**{**

**temp=temp->next;**

**}**

**printf("\n The previous ele of %d is %d\n",s,temp->data);**

**}**

**}**

**void find(int s)**

**{**

**struct node \*temp; temp=head; if(head==NULL)**

**{**

**printf("\n List is empty");**

**}**

**else**

**{**

**while(temp->data!=s && temp->next!=NULL)**

**{**

**temp=temp->next;**

**}**

**if(temp!=NULL && temp->data==s)**

**{**

**printf("\n Searching ele %d is present in the addr of %p",temp-**

**>data,temp);**

**}**

**else**

**{**

**printf("\n Searching elem %d is not present",s);**

**}**

**}**

**}**

**void isempty()**

**{**

**if(head==NULL)**

**{**

**printf("\nList is empty\n");**

**}**

**else**

**{**

**printf("\nList is not empty\n");**

**}**

**}**

**void deleteAtBeginning()**

**{**

**struct node \*t; t=head;**

**head=t->next;**

**}**

**void deleteAtEnd()**

**{**

**struct node \*temp; temp=head; if(head==NULL)**

**{**

**printf("\n List is empty");**

**}**

**else**

**{**

**while(temp->next->next!=NULL)**

**{**

**temp=temp->next;**

**}**

**temp->next=NULL;**

**}**

**}**

**void display()**

**{**

**struct node \*t; t=head; while(t!=NULL)**

**{**

**printf("%d\t",t->data); t=t->next;**

**}**

**}**

**void delete(int ele)**

**{**

**struct node \*t; t=head;**

**if(t->data==ele)**

**{**

**head=t->next;**

**}**

**else**

**{**

**while(t->next->data!=ele)**

**{**

**t=t->next;**

**}**

**t->next=t->next->next;**

**}**

**}**

**int main()**

**{**

**do**

**{**

**int ch,a,pos;**

**printf("\n Choose any one operation that you would like to perform\n"); printf("\n 1.Insert the element at the beginning");**

**printf("\n 2.Insert the element at the end"); printf("\n 3. To insert at the specified position"); printf("\n 4. To view list");**

**printf("\n 5.To view list size"); printf("\n 6.To delete first element"); printf("\n 7.To delete last element"); printf("\n 8.To find next element"); printf("\n 9. To find previous element");**

**printf("\n 10. To find search for an element"); printf("\n 11. To quit");**

**printf("\n Enter your choice\n"); scanf("%d",&ch);**

**switch(ch)**

**{**

**case 1:**

**printf("\n Insert an element to be inserted at the beginning\n"); scanf("%d",&a);**

**insertfront(a); break;**

**case 2:**

**printf("\n Insert an element to be inserted at the End\n"); scanf("%d",&a);**

**insertend(a); break;**

**case 3:**

**printf("\n Insert an element and the position to insert in the list\n"); scanf("%d%d",&a,&pos);**

**insertpos(a,pos); break;**

**case 4:**

**display(); break; case 5:**

**listsize(); break; case 6:**

**printf("\n Delete an element to be in the beginning\n"); deleteAtBeginning();**

**break; case 7:**

**printf("\n Delete an element to be at the end\n"); deleteAtEnd();**

**break;**

**case 8:**

**printf("\n enter the element to which you need to find next ele in the list\n");;**

**scanf("%d",&a); findnext(a); break;**

**case 9:**

**printf("\n enter the element to which you need to find prev ele in the list\n");**

**scanf("%d",&a); findprev(a); break;**

**case 10:**

**printf("\n enter the element to find the address of it\n"); scanf("%d",&a);**

**find(a); break; case 11:**

**printf("Ended"); exit(0); default:**

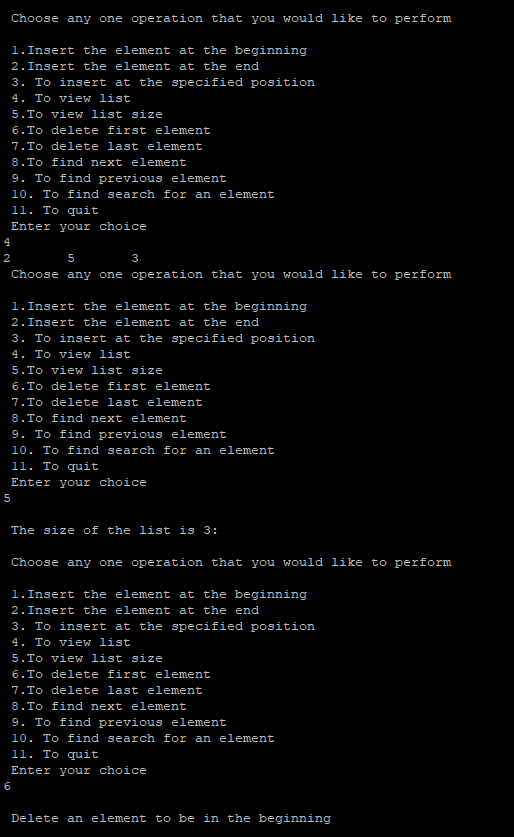
**printf("Invalid option is chosen so the process is quit");**

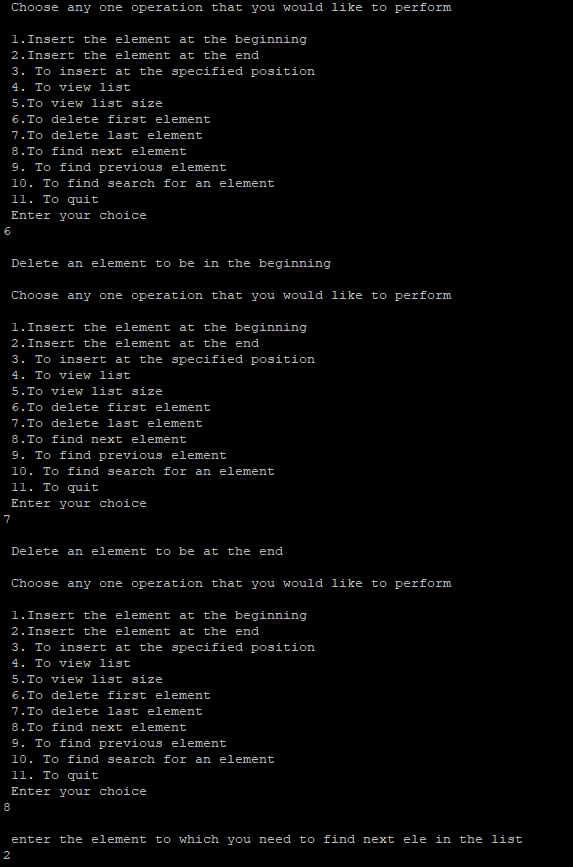
**}**

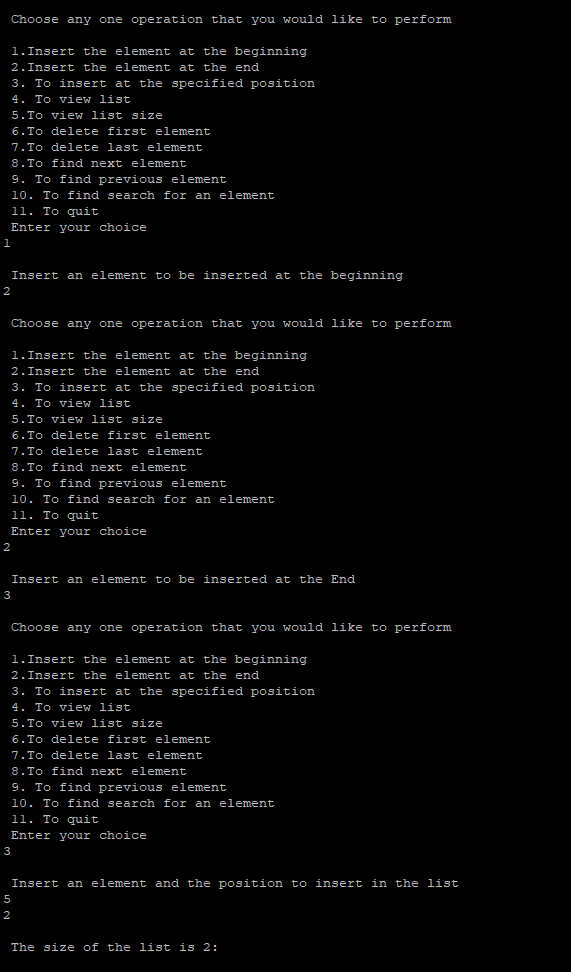
**}while(1); return 0;**

**}**

**OUTPUT:**







**RESULT: Thus the program was successfully executed**