**Objective –**

Write a program to implement singly linked list with different operation.

A. Creation

B. Inseration (3)- at starting,at end ,after some node….

C. deletion (3)- at starting,at end ,after some node….

D. Traversing

**Description –**

Singly linked list:-

A linked-list is a sequence of data structures which are connected together via links.

Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list the second most used data structure after array. Following are important terms to understand the concepts of Linked List.

This linked list has sequential one-way connection with adjacent nodes. It can only be parsed one-way.

Two important points to remember:

* head points to the first node of the linked list
* next pointer of the last node is NULL, so if the next current node is NULL, we have reached the end of the linked list.

In all of the examples, we will assume that the linked list has three nodes 1 --->2 --->3 with node structure as below:

struct node

{

int data;

struct node \*next;

};

Operation of linked list :-

1.insertion():-adds an element at the beginning,end and at given position.

2.deletion():-deletes a element at beginning ,end and at given location.

3.create():- it creates new node/linked list.

4.traverse():-it displays element present in node/linked list.

2Function: -

* Function is a block of code that performs a particular task.
* Every c program can be thought of as a collection of these functions.
* Function name must be unique.
* Function has name for identification.
* Important parts of function – written type, function call, function declaration, function definition
* A c program is a collection of one or more functions.
* There is no limit on the number of functions that might be present in a c program.
* After each function has done its thing, control returns to main ().

**Program –**

#include<stdio.h> // header files

#include<stdlib.h>

#include<conio.h>

void create(); // function declaration

void display(); // function declaration

void insert\_begin(); // function declaration

void insert\_end(); // function declaration

void insert\_pos(); // function declaration

void delete\_begin(); // function declaration

void delete\_end(); // function declaration

void delete\_pos(); // function declaration

struct node

{

int info;

struct node \*next;

};

struct node \*start=NULL;

int main() //main() starts

{

int choice;

while(1){ //while loop

printf("\n\*\*\*SINGLE LINKED LIST OPERATIONS:\*\*\*\*\n");

printf("\n MENU \n");

printf("---------------------------------------\n");

printf("\n 1.Create \n");

printf("\n 2.Display \n");

printf("\n 3.Insert at the beginning \n");

printf("\n 4.Insert at the end \n");

printf("\n 5.Insert at specified position \n");

printf("\n 6.Delete from beginning \n");

printf("\n 7.Delete from the end \n");

printf("\n 8.Delete from specified position \n");

printf("\n 9.Exit \n");

printf("\n--------------------------------------\n");

printf("Enter your choice:\t");

scanf("%d",&choice);

switch(choice) // switch statement

{

case 1:

create(); //function calling

break; //break statement

case 2:

display(); //function calling

break;

case 3:

insert\_begin(); //function calling

break;

case 4:

insert\_end(); //function calling

break;

case 5:

insert\_pos(); //function calling

break;

case 6:

delete\_begin(); //function calling

break;

case 7:

delete\_end(); //function calling

break;

case 8:

delete\_pos(); //function calling

break;

case 9:

exit(0);

break;

default:

printf("\n Wrong Choice:\n");

break;

}//end of switch()

}

return 0;

}//end of main()

void create() //function definition

{

struct node \*temp,\*ptr;

temp=(struct node \*)malloc(sizeof(struct node));

if(temp==NULL)

{

printf("\nOut of Memory Space:\n");

exit(0);

}

printf("\nEnter the data value for the node:\t");

scanf("%d",&temp->info);

temp->next=NULL;

if(start==NULL)

{

start=temp;

}

else

{

ptr=start;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=temp;

}

}//end of create()

void display() //function definition

{

struct node \*ptr;

if(start==NULL)

{

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

return;

}

else

{

ptr=start;

printf("\nThe List elements are:\n");

while(ptr!=NULL)

{

printf("%d\t",ptr->info );

ptr=ptr->next ;

}//end of while

}//end of else

}//end of display()

void insert\_begin() //function definition

{

struct node \*temp;

temp=(struct node \*)malloc(sizeof(struct node));

if(temp==NULL)

{

printf("\nOut of Memory Space:\n");

return;

}

printf("\nEnter the data value for the node:\t" );

scanf("%d",&temp->info);

temp->next =NULL;

if(start==NULL)

{

start=temp;

}

else

{

temp->next=start;

start=temp;

}

}//end of insert\_begin()

void insert\_end() //function definition

{

struct node \*temp,\*ptr;

temp=(struct node \*)malloc(sizeof(struct node));

if(temp==NULL)

{

printf("\nOut of Memory Space:\n");

return;

}

printf("\nEnter the data value for the node:\t" );

scanf("%d",&temp->info );

temp->next =NULL;

if(start==NULL)

{

start=temp;

}

else

{

ptr=start;

while(ptr->next !=NULL)

{

ptr=ptr->next ;

}

ptr->next =temp;

}

}//end of insert\_end

void insert\_pos() //function definition

{

struct node \*ptr,\*temp;

int i,pos;

temp=(struct node \*)malloc(sizeof(struct node));

if(temp==NULL)

{

printf("\nOut of Memory Space:\n");

return;

}

printf("\nEnter the position for the new node to be inserted:\t");

scanf("%d",&pos);

printf("\nEnter the data value of the node:\t");

scanf("%d",&temp->info) ;

temp->next=NULL;

if(pos==0)

{

temp->next=start;

start=temp;

}

else

{

for(i=0,ptr=start;i<pos-1;i++)

{

ptr=ptr->next;

if(ptr==NULL)

{

printf("\nPosition not found:[Handle with care]\n");

return;

}

}

temp->next =ptr->next ;

ptr->next=temp;

}//end of else

}//end of insert\_pos

void delete\_begin() //function definition

{

struct node \*ptr;

if(ptr==NULL)

{

printf("\nList is Empty:\n");

return;

}

else

{

ptr=start;

start=start->next ;

printf("\nThe deleted element is :%d\t",ptr->info);

free(ptr);

}

}//end of delete\_begin()

void delete\_end() //function definition

{

struct node \*temp,\*ptr;

if(start==NULL)

{

printf("\nList is Empty:");

exit(0);

}

else if(start->next ==NULL)

{

ptr=start;

start=NULL;

printf("\nThe deleted element is:%d\t",ptr->info);

free(ptr);

}

else

{

ptr=start;

while(ptr->next!=NULL)

{

temp=ptr;

ptr=ptr->next;

}

temp->next=NULL;

printf("\n the deleted element is %d\t",ptr->info);

free(ptr);

}

}

void delete\_pos() //function definition

{

int i,pos;

struct node \*temp,\*ptr;

if(start==NULL)

{

printf("\n the list is empty:\n");

exit(0);

}

else

{

printf("\n enter the position of the node to be deleted:\t");

if(pos==0)

{

ptr=start;

start=start->next;

printf("\nthe deleted element is:%d\t",ptr->info);

free(ptr);

}

else

{

ptr=start;

for(i=0;i<pos;i++)

{

temp=ptr;

ptr=ptr->next;

if(ptr==NULL)

{

printf("\nposition not found:\n");

return;

}

}

temp->next=ptr->next;

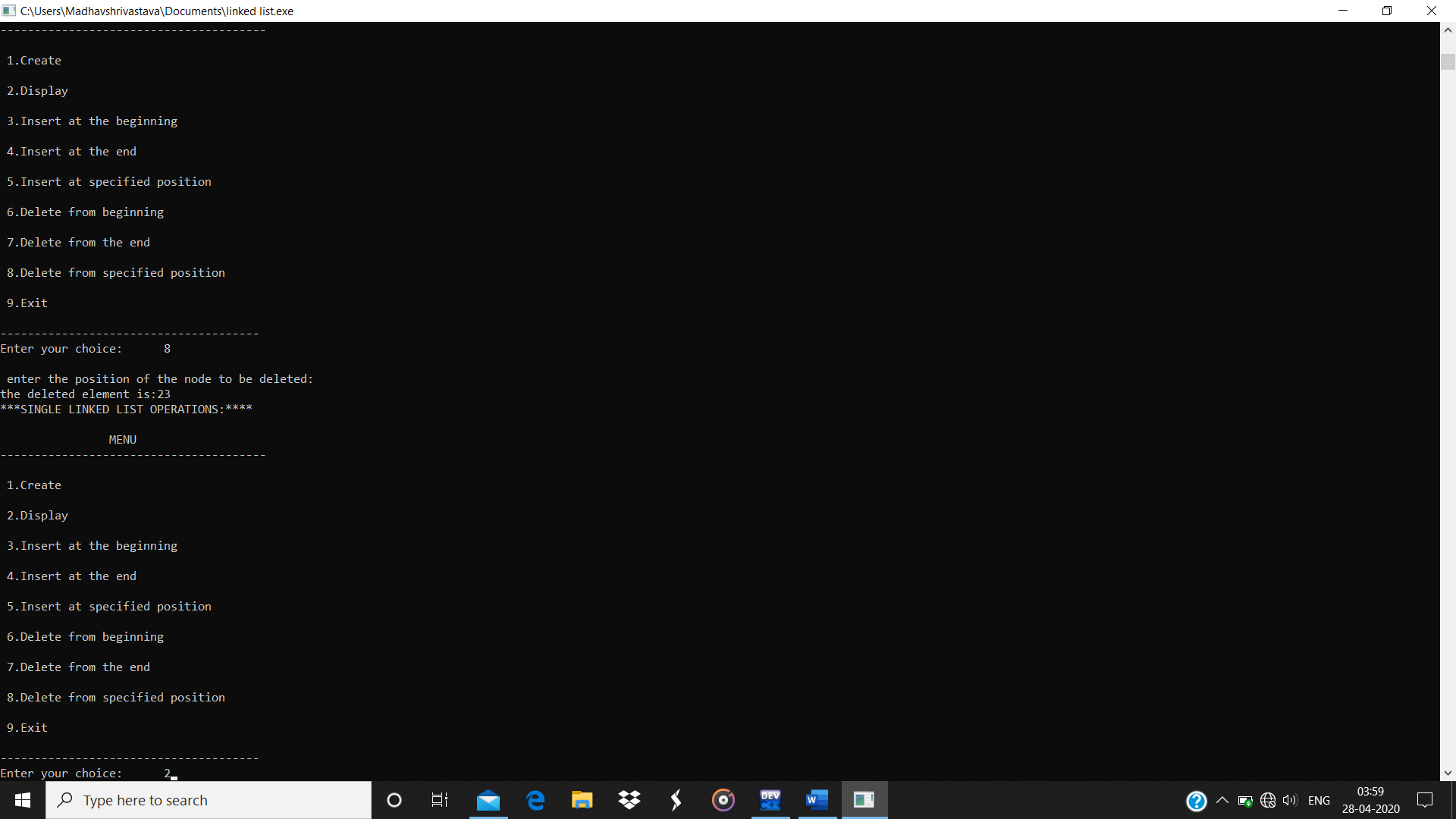
printf("\n the deleted elements is:%d\t",ptr->info);

free(ptr);

}

}

}

Output:

