**NAME:Abhishek Tiwari**

**BRANCH:COMPS-3**

**BATCH:C**

**ROLL NO:55**

**Aim:** Implementation of Singly Linked List

**Objective:** It is used to implement stacks and queue which are linked needs throughout computer science .To prevent the Collision between the data in the Hash map.we use a singly Linked list

**Theory:**

Singly linked lists contain nodes which have a data part as well as an address part i.e. next, which points to the next node in sequence of nodes. The operations we can perform on singly linked lists are insertion, deletion and traversal.

**Algorithm:**

Traversing a linked list:

Step 1: take a temp node.

Step 2: traversal a temp node not equal to NULL.

Step 3: printf the data.

Inserting at beginning:

Step 1: take a newNode.

Step 2: create a dynamic allocation for newNode.

Step 3: insert a data for newNode.

Step 4: set newNode-> next = head.

Step 5: and head = newNode;

Deleting a node at beginning:

Step 1: take a temp node.

Step 2: set the value of temp equal to head.

Step 3: head = head->next.

Step 4: free temp form memory location.

Step 5: return head.

**Code**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct Node

{

int data;

struct Node \*next;

};

void linklistTraversal(struct Node \*ptr)

{

while(ptr!=NULL)

{

printf("Elements: %d\n",ptr->data);

ptr = ptr->next;

}

}

struct Node \*deleteBegining(struct Node \*head)

{

struct Node \*ptr = head;

head = head->next;

free(ptr);

return head;

}

struct Node \*insertionBegining(struct Node \*ptr)

{

struct Node \*newNode;

newNode = (struct Node \*)malloc(sizeof(struct Node));

printf("Enter the data");

scanf("%d",&newNode->data);

newNode->next = ptr;

return newNode;

}

int main()

{

int choice;

struct Node \*head;

struct Node \*first;

struct Node \*second;

struct Node \*third;

clrscr();

head =(struct Node \*)malloc(sizeof(struct Node));

first = (struct Node \*)malloc(sizeof(struct Node));

second = (struct Node \*)malloc(sizeof(struct Node));

third = (struct Node \*)malloc(sizeof(struct Node));

head->data = 24;

head->next = first;

first->data = 34;

first->next = second;

second->data = 57;

second->next = third;

third->data = 89;

third->next = NULL;

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

printf("1.Traversal\n2.Insertion at Begining\n3.Deltion at Beigining\n4.Exit\n");

do

{

printf("Enter your choice: ");

scanf("%d",&choice);

switch(choice)

{

case 1:

linklistTraversal(head);

break;

case 2:

head = insertionBegining(head);

linklistTraversal(head);

break;

case 3:

head = deleteBegining(head);

linklistTraversal(head);

break;

case 4:

break;

default:

printf("Invlid option\n");

break;

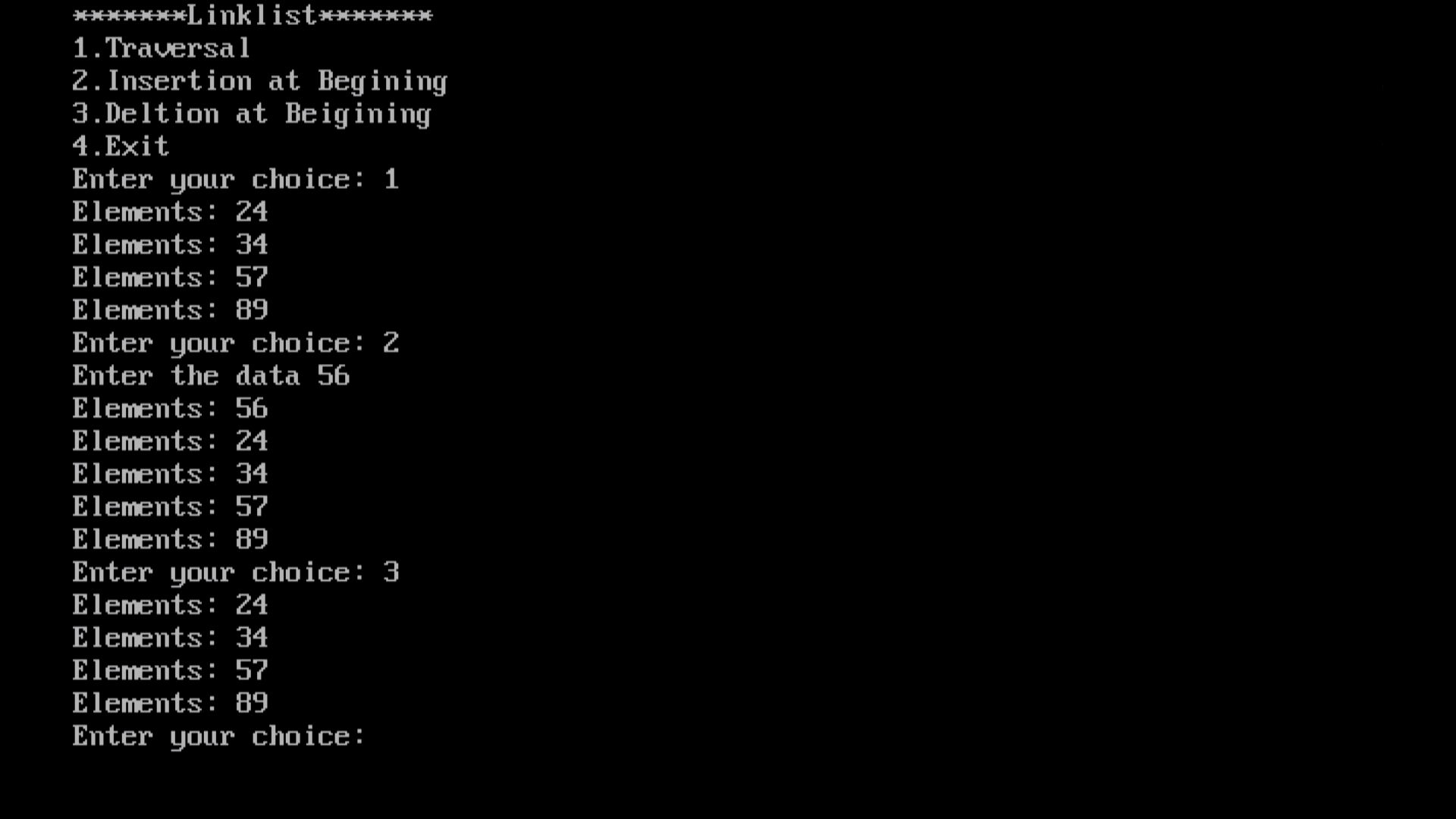
}

}while(choice!=4);

getch();

return 0;

}

**Output :**

**Conclusion :**

1. Insertion and deletion of node are easier.
2. Data structure such as stack and queues can be easily implemented using linked list.

**Ans**

**This C implementation of a singly linked list is well-structured and user-friendly, with meaningful names and proper memory management. It covers essential operations effectively, but error handling for memory allocation and non-standard headers could be improved for better code reliability.**