**NAME: ASHWINI SHIVAJI RATHOD**

**CLASS:SE**

**DIV:3**

**Experiment no 6**

**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 :**

1)Node Structure (struct Node): This structure defines a node in the singly linked list. Each node has an integer data (Data) and a pointer to the next node (next).

2)Insertion at the Start (insertStart function): This function inserts a new node with the specified data at the beginning of the linked list. It allocates memory for a new node, sets its data and next pointer to the current head, and updates the head to point to the new node.

3)Deletion from the Start (deleteStart function): This function deletes the node at the beginning of the linked list. It first checks if the list is empty. If not, it updates the head to point to the second node (if it exists) and frees the memory of the deleted node.

4)Traversal (display function): This function is used to traverse and print the elements of the singly linked list. It starts at the head of the list and iterates through the nodes, printing their data until it reaches the end (i.e., a NULL node).

**Algorithm:**

**1)Node Structure:**

struct Node

{

int Data;

struct Node \*next;

};

**2)Insertion at the Start:**

Function insertStart (head, data):

1. Create a new node newNode.

2. Set newNode's Data to data.

3. Set newNode's next pointer to the current head.

4. Update the head pointer to newNode.

**3)Deletion from the Start:**

Function deleteStart (head):

1. Initialize a temporary pointer temp to the current head.

2. If head is NULL, print "Linked List Empty, nothing to delete," and return.

3. Update the head pointer to point to the next node (head->next).

4. Free the memory occupied by the temp node.

**4)Traversal:**

Function display (head):

1. Print "Linked List: ".

2. Initialize a pointer node to the head of the list.

3. While node is not NULL:

a. Print the data in the current node.

b. Move to the next node by updating node to node->next.

4. Print a newline to end the list display.

**Code :**

The syntax for creating a node

struct Node

{

int Data;

Struct Node \*next;

};

## Insertion of a node

void insertStart (struct Node \*\*head, int data)

{

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

newNode - >

data = data;

newNode - >

next = \*head;

//changing the new head to this freshly entered node

\*head = newNode;

}

## Deletion of a node

void deleteStart(struct Node \*\*head)

{

struct Node \*temp = \*head;

// if there are no nodes in Linked List can't delete

if (\*head == NULL)

{

printf ("Linked List Empty, nothing to delete");

return;

}

// move head to next node

\*head = (\*head)->next;

free (temp);}

## Traversal in a Singly Linked List

void display(struct Node\* node)

{

printf("Linked List: ");

// as linked list will end when Node is Null

while(node!=NULL){

printf("%d ",node->data);

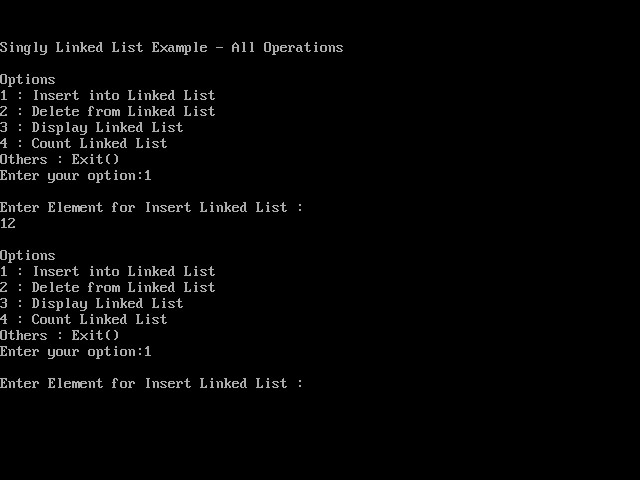
node = node->next;

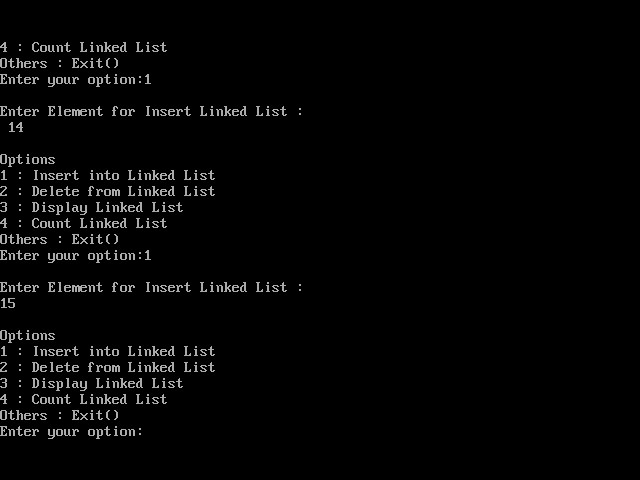
}

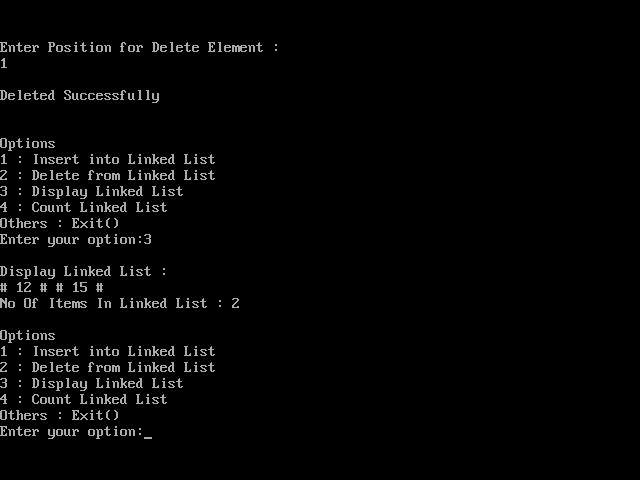
printf("\n");

}

**Output :**







**Conclusion :**

The provided C code implements a basic singly linked list. It includes functions for inserting nodes at the beginning, deleting nodes from the beginning, and displaying the list. This code serves as a fundamental example for working with singly linked lists, a fundamental data structure in computer science.