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:**

* Linked List can be defined as collection of objects called nodes that are randomly stored in the memory.
* A node contains two fields i.e. data stored at that particular address and the pointer which contains the address of the next node in the memory.
* The last node of the list contains pointer to the null

**Algorithm:**

Inserting a node at the beginning

Algorithm:

Step 1: SET NEW\_NODE = PTR

Step 2: SET PTR = PTR → NEXT

Step 3: SET NEW\_NODE → DATA = VAL

Step 4: SET NEW\_NODE → NEXT = HEAD

Step 5: SET HEAD = NEW\_NODE

Step 6: EXIT

Inserting a node at the ending

Step 1: SET NEW\_NODE = PTR

Step 2: SET PTR = PTR - > NEXT

Step 3: SET NEW\_NODE - > DATA = VAL

Step 4: SET NEW\_NODE - > NEXT = NULL

Step 5: SET PTR = HEAD

Step 6: Repeat Step 7 while PTR - > NEXT != NULL

Step 7: SET PTR = PTR - > NEXT

[END OF LOOP]

Step 8: SET PTR - > NEXT = NEW\_NODE

Step 9: EXIT

Inserting a node at Middle

struct node \*newNode;

newNode = malloc(sizeof(struct node));

newNode->data = 4;

struct node \*temp = head;

for(int i=2; i < position; i++) {

if(temp->next != NULL) {

temp = temp->next;

}

}

newNode->next = temp->next;

temp->next = newNode;

## Deletion of a node

Step 1: IF START = NULL Write UNDERFLOW Go to Step 5 [END OF IF]

Step 2: SET PTR = START

Step 3: SET START = START NEXT

Step 4: FREE PTR

Step 5: EXIT

**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);

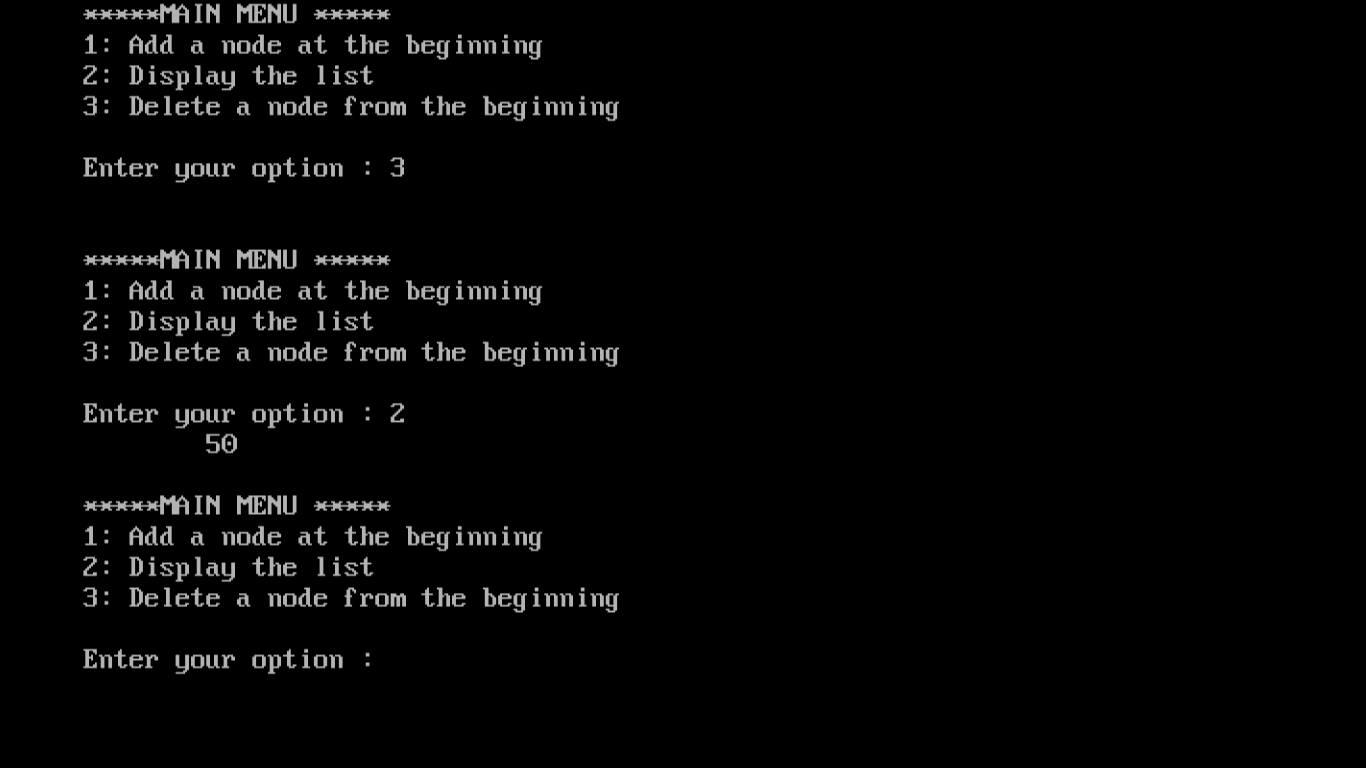
node = node->next;

}

printf("\n");

}

**Output**:



**Conclusion:** Hence implementation of singly linked list is verified.