**NAME - SUVIDHI V. PAREEK**

**SE(COMPS) / DIV - 3 / ROLL NO. 10**

**Experiment No. 6: Singly Linked List Operations**

**Aim:** Implementation of Singly Linked List

**Objective:**

It is used to implement stacks and queues which are like fundamental needs throughout computer science. To prevent the collision between the data in the hash map, we use a singly linked list.

**THEORY:**

A Singly linked list is a ordered collection of finite, homogenous elements referred as a node. Each node consist of two fields: one field for data which is referred as information and other field is an address field to store the address of next element in the list.

The address field of last node contains null value to indicate the end of list. The elements of linked list are not stored in continuous memory location but they are scattered, and still bounded to each other by an explicit link

Header is a node containing null in its information field and an next address field contains the address of first data node in the list.

**Algorithm**

Algorithm : INSERT\_SPECIFIED(Header, X, Key)

Input : Header is a pointer to header node. X is a data of node to be inserted and Key is data of node after which insertion is to be done.

Output :A singly linked list enriched with newly inserted node.

Data Structure : A singly linked list whose address of starting node is in Header. And two fields info and next to point to data field and address field respectively. ptr is used to an address of current node

1. new = GETNODE()
2. if new = NULL then

print “Insufficient Memory”

Exit

3. else

ptr = HEADER

while info(ptr)!= Key AND next(ptr)!= NULL do

ptr = next(ptr)

end while

1. if next(ptr) = NULL then

print “Key not found”

exit

1. else

next(new) = next(ptr)

info(new) = x

next(ptr) = new

end if

end if

1. stop

Algorithm : DELETE\_SPECIFIED(Header, Key)

Input : Header is a pointer to header node. Key is data of node after which is to be deleted.

Output :A singly linked list with removed node.

Data Structure : A singly linked list whose address of starting node is in Header. And two fields info and next to point to data field and address field respectively. ptr is used to an address of current node

1. ptr1 = HEADER

ptr = next(ptr1)

1. while ptr!= NULL do

if info(ptr) != Key

ptr1 = ptr

ptr = next(ptr)

else

next(ptr1) = next(ptr)

print(info(ptr))

FREENODE(ptr)

End if

End while

1. if ptr = NULL then

print “ key not found”

end if

1. stop

Algorithm : TRAVERSAL(Header)

Input : Header is a pointer to header node.

Output :A singly linked list is traversed and its data value is printed.

Data Structure : A singly linked list whose address of starting node is in Header. And two fields info and next to point to data field and address field respectively. ptr is used to an address of current node

1. ptr = next(HEADER)
2. while ptr!= NULL do

print (Info(ptr))

End while

1. stop

**Code:**

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

struct Node\* createNode(int data) {

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

if (newNode == NULL) {

printf("Memory allocation failed.\n");

exit(1);

}

newNode->data = data;

newNode->next = NULL;

return newNode;

}

void insertEnd(struct Node\*\* head, int data) {

struct Node\* newNode = createNode(data);

if (\*head == NULL) {

\*head = newNode;

} else {

struct Node\* current = \*head;

while (current->next != NULL) {

current = current->next;

}

current->next = newNode;

}

}

void deleteNode(struct Node\*\* head, int key) {

struct Node\* current = \*head;

struct Node\* prev = NULL;

if (current != NULL && current->data == key) {

\*head = current->next;

free(current);

return;

}

while (current != NULL && current->data != key) {

prev = current;

current = current->next;

}

if (current == NULL) {

printf("Node with value %d not found.\n", key);

return;

}

prev->next = current->next;

free(current);

}

void traverseList(struct Node\* head) {

struct Node\* current = head;

printf("Linked List: ");

while (current != NULL) {

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

current = current->next;

}

printf("NULL\n");

}

int main() {

struct Node\* head = NULL; //

int n, data, key;

printf("Enter the number of elements to insert: ");

scanf("%d", &n);

for (int i = 0; i < n; i++) {

printf("Enter element %d: ", i + 1);

scanf("%d", &data);

insertEnd(&head, data);

}

traverseList(head);

printf("Enter the value to delete: ");

scanf("%d", &key);

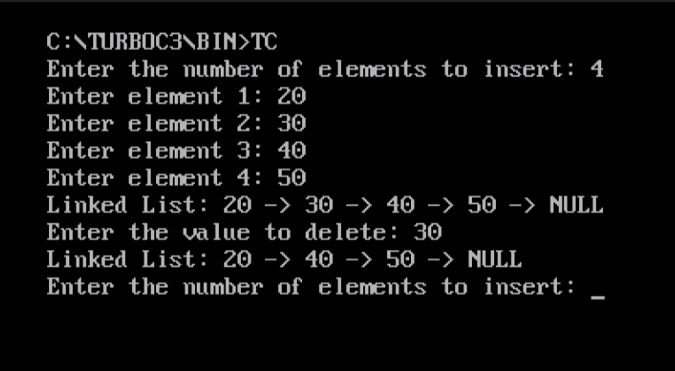
deleteNode(&head, key);

traverseList(head);

return 0;

}

**Output:**



**Conclusion:**

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.Various operations can be performed on singly linked list like insertion at front, end and at specified position , deletion at front, end and at specified position, traversal, copying and merging.