**Experiment – 4**

**Single linked List**

*Source Code:*

#include <stdio.h>

#include <stdlib.h>

// Define the structure for a singly linked list node

struct Node {

int data;

struct Node\* next;

};

// Function to insert a new node at the beginning of the linked list

struct Node\* insertAtBeginning(struct Node\* head, int data) {

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

newNode->data = data;

newNode->next = head;

return newNode;

}

// Function to insert a new node at the end of the linked list

struct Node\* insertAtEnd(struct Node\* head, int data) {

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

newNode->data = data;

newNode->next = NULL;

if (head == NULL) {

return newNode;

}

struct Node\* current = head;

while (current->next != NULL) {

current = current->next;

}

current->next = newNode;

return head;

}

// Function to insert a new node after a given node

void insertAfterNode(struct Node\* prevNode, int data) {

if (prevNode == NULL) {

printf("Previous node cannot be NULL.\n");

return;

}

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

newNode->data = data;

newNode->next = prevNode->next;

prevNode->next = newNode;

}

// Function to delete the first node of the linked list

struct Node\* deleteAtBeginning(struct Node\* head) {

if (head == NULL) {

printf("List is empty. Cannot delete.\n");

return head;

}

struct Node\* temp = head;

head = head->next;

free(temp);

return head;

}

// Function to delete the last node of the linked list

struct Node\* deleteAtEnd(struct Node\* head) {

if (head == NULL) {

printf("List is empty. Cannot delete.\n");

return head;

}

if (head->next == NULL) {

free(head);

return NULL;

}

struct Node\* current = head;

while (current->next->next != NULL) {

current = current->next;

}

free(current->next);

current->next = NULL;

return head;

}

// Function to print the linked list

void printList(struct Node\* head) {

struct Node\* current = head;

while (current != NULL) {

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

current = current->next;

}

printf("NULL\n");

}

int main() {

struct Node\* head = NULL;

struct Node\* current = NULL;

int data, numNodes;

printf("Enter the number of nodes: ");

scanf("%d", &numNodes);

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

printf("Enter data for node %d: ", i + 1);

scanf("%d", &data);

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

newNode->data = data;

newNode->next = NULL;

if (head == NULL) {

head = newNode;

current = newNode;

} else {

current->next = newNode;

current = newNode;

}

}

// Print the linked list

printf("Linked List: ");

printList(head);

// Insertion operations

int insertChoice, insertData;

printf("\nInsertion Menu:\n");

printf("1. Insert at beginning\n");

printf("2. Insert at end\n");

printf("3. Insert after a given node\n");

printf("Enter your choice: ");

scanf("%d", &insertChoice);

switch (insertChoice) {

case 1:

printf("Enter data to insert at the beginning: ");

scanf("%d", &insertData);

head = insertAtBeginning(head, insertData);

break;

case 2:

printf("Enter data to insert at the end: ");

scanf("%d", &insertData);

head = insertAtEnd(head, insertData);

break;

case 3:

printf("Enter data to insert: ");

scanf("%d", &insertData);

int afterNodeData;

printf("Enter data of the node after which to insert: ");

scanf("%d", &afterNodeData);

struct Node\* current = head;

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

current = current->next;

}

if (current == NULL) {

printf("Node with given data not found.\n");

} else {

insertAfterNode(current, insertData);

}

break;

default:

printf("Invalid choice for insertion.\n");

break;

}

// Print the linked list after insertion

printf("Linked List after insertion: ");

printList(head);

// Deletion operations

int deleteChoice;

printf("\nDeletion Menu:\n");

printf("1. Delete at beginning\n");

printf("2. Delete at end\n");

printf("Enter your choice: ");

scanf("%d", &deleteChoice);

switch (deleteChoice) {

case 1:

head = deleteAtBeginning(head);

break;

case 2:

head = deleteAtEnd(head);

break;

default:

printf("Invalid choice for deletion.\n");

break;

}

// Print the linked list after deletion

printf("Linked List after deletion: ");

printList(head);

// Free memory by deleting the nodes

current = head;

while (current != NULL) {

struct Node\* temp = current;

current = current->next;

free(temp);

}

return 0;

}

*Output:*

Enter the number of nodes: 3

Enter data for node 1: 1

Enter data for node 2: 2

Enter data for node 3: 3

Linked List: 1 -> 2 -> 3 -> NULL

Insertion Menu:

1. Insert at beginning

2. Insert at end

3. Insert after a given node

Enter your choice: 1

Enter data to insert at the beginning: 0

Linked List after insertion: 0 -> 1 -> 2 -> 3 -> NULL

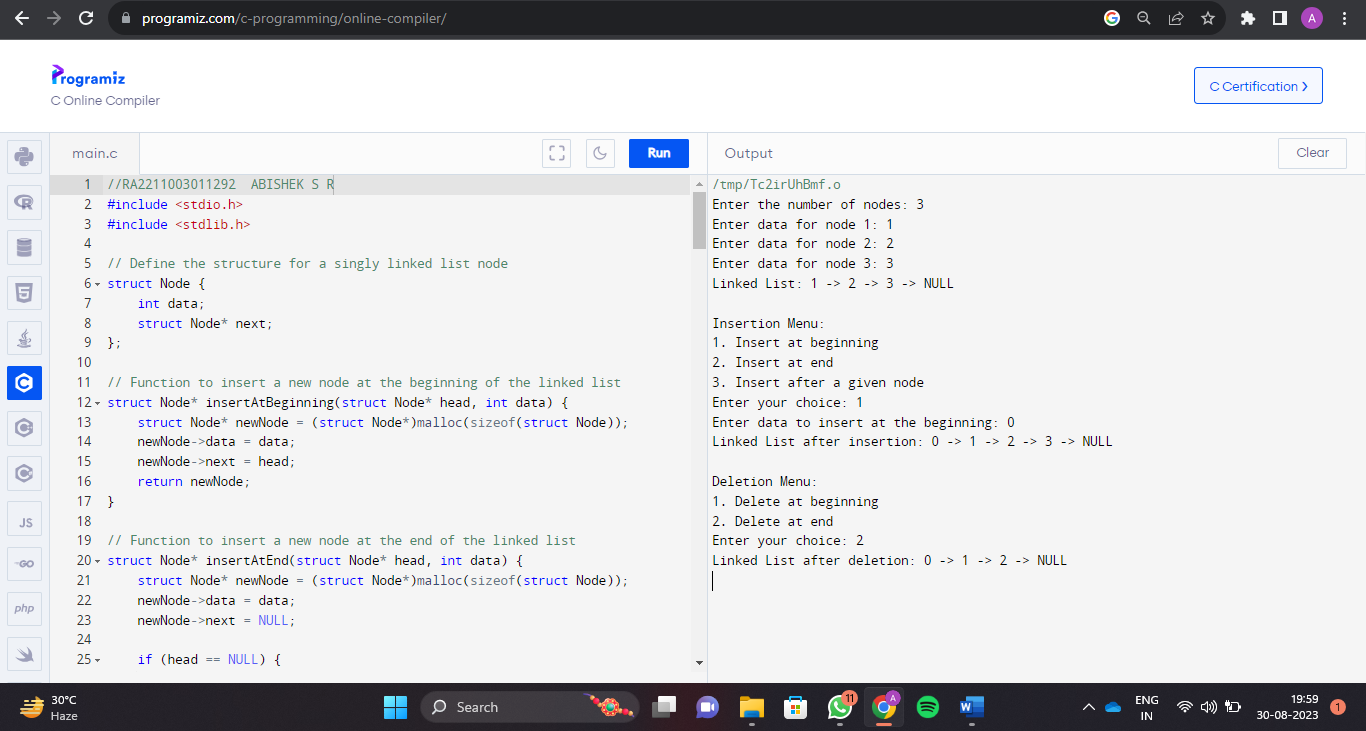
Deletion Menu:

1. Delete at beginning

2. Delete at end

Enter your choice: 2

Linked List after deletion: 0 -> 1 -> 2 -> NULL



**ABISHEK S R**

**RA2211003011292**