Name:Shakib Ahmad Shaikh

S.E Comps:3 Roll:no.37 Batch-B

Experiment No 6:

Aim : Implementation of Singly Linked List

Algorithm: Insert at Beginning

Input: head (pointer to the head of the linked list), data (integer to insert)

Output: Updated head (if the head changes)

1. Create a new node newNode with data.

2. Set newNode->next to the current head.

3. Update the head to newNode.

4. Return the updated head.

Algorithm: Insert at End

Input: head (pointer to the head of the linked list), data (integer to insert)

Output: None (the linked list is modified in place)

1. Create a new node newNode with data.

2. If the list is empty (head is NULL), set the head to newNode and return.

3. Initialize a current pointer to head.

4. Traverse the list until current->next is NULL.

5. Set current->next to newNode.

Algorithm: Delete a Node

Input: head (pointer to the head of the linked list), data (integer to delete)

Output: Updated head (if the head changes)

1. If the list is empty (head is NULL), return head.

2. If the node to delete is the head, set head to head->next and free the old head.

3. Initialize a current pointer to head and a previous pointer to NULL.

4. Traverse the list:

a. While current is not NULL and current->data is not equal to data:

i. Set previous to current.

ii. Set current to current->next.

b. If current is NULL, the data was not found, so return head.

c. Otherwise, set previous->next to current->next and free current.

5. Return the updated head.

Algorithm: Display the List

Input: head (pointer to the head of the linked list)

Output: None (prints the elements of the list)

1. Initialize a current pointer to head.

2. While current is not NULL:

a. Print current->data.

b. Set current to current->next.

3. End.

Algorithm: Exit Program

Input: None

Output: None

1. Exit the program gracefully.

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;

}

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

struct Node\* newNode = createNode(data);

newNode->next = head;

return newNode;

}

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

struct Node\* newNode = createNode(data);

if (head == NULL) {

return newNode;

}

struct Node\* current = head;

while (current->next != NULL) {

current = current->next;

}

current->next = newNode;

return head;

}

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

if (head == NULL) {

printf("List is empty. Nothing to delete.\n");

return head;

}

if (head->data == data) {

struct Node\* temp = head;

head = head->next;

free(temp);

return head;

}

struct Node\* current = head;

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

current = current->next;

}

if (current->next == NULL) {

printf("Element %d not found in the list.\n", data);

return head;

}

struct Node\* temp = current->next;

current->next = current->next->next;

free(temp);

return head;

}

void displayList(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 choice, data;

while (1) {

printf("\nSingly Linked List Operations:\n");

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

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

printf("3. Delete a node\n");

printf("4. Display the list\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

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

scanf("%d", &data);

head = insertAtBeginning(head, data);

break;

case 2:

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

scanf("%d", &data);

head = insertAtEnd(head, data);

break;

case 3:

printf("Enter data to delete: ");

scanf("%d", &data);

head = deleteNode(head, data);

break;

case 4:

displayList(head);

break;

case 5:

exit(0);

default:

printf("Invalid choice. Please try again.\n");

}

}

return 0;

}

Output:

Singly Linked List Operations:

1. Insert at the beginning

2. Insert at the end

3. Delete a node

4. Display the list

5. Exit

Enter your choice: 3

Enter data to delete: 4

List is empty. Nothing to delete.

Singly Linked List Operations:

1. Insert at the beginning

2. Insert at the end

3. Delete a node

4. Display the list

5. Exit

Enter your choice: 5