**LAB – 7**

**Name : Abhinav Sanjay**

**USN : 1BM23CS009**

**WAP to Implement Single Link List with following operations: Sort the linked list, Reverse the linked list, Concatenation of two linked lists.**

#include <stdio.h>

#include <stdlib.h>

struct node {

int value;

struct node \*next;

};

typedef struct node\* NODE;

NODE getnode() {

NODE new\_node = (NODE)malloc(sizeof(struct node));

if (new\_node == NULL) {

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

exit(1);

}

new\_node->next = NULL;

return new\_node;

}

NODE insert\_end(int item, NODE first) {

NODE new\_end = getnode();

new\_end->value = item;

new\_end->next = NULL;

if (first == NULL) {

return new\_end;

}

NODE current = first;

while (current->next != NULL) {

current = current->next;

}

current->next = new\_end;

return first;

}

NODE reverse(NODE first) {

NODE current = NULL;

NODE temp;

while (first != NULL) {

temp = first;

first = first->next;

temp->next = current;

current = temp;

}

return current;

}

NODE concatenate(NODE first\_1, NODE first\_2) {

if (first\_1 == NULL) {

return first\_2;

}

if (first\_2 == NULL) {

return first\_1;

}

NODE last1 = first\_1;

while (last1->next != NULL) {

last1 = last1->next;

}

last1->next = first\_2;

return first\_1;

}

NODE sort(NODE first) {

if (first == NULL || first->next == NULL) {

return first;

}

NODE temp1, temp2;

int temp\_value;

for (temp1 = first; temp1 != NULL; temp1 = temp1->next) {

for (temp2 = temp1->next; temp2 != NULL; temp2 = temp2->next) {

if (temp1->value > temp2->value) {

// Swap values

temp\_value = temp1->value;

temp1->value = temp2->value;

temp2->value = temp\_value;

}

}

}

return first;

}

void display(NODE first) {

if (first == NULL) {

printf("Linked list is empty.\n");

return;

}

NODE temp = first;

while (temp != NULL) {

printf("%d\t", temp->value);

temp = temp->next;

}

printf("\n");

}

int main() {

NODE first\_1 = NULL;

NODE first\_2 = NULL;

int choice, item, pos;

while (1) {

printf("\nMenu:\n");

printf("1. Insert in linked list 1\n");

printf("2. Insert in linked list 2\n");

printf("3. Sort in linked list 1\n");

printf("4. Sort in linked list 2\n");

printf("5. Reverse in linked list 1\n");

printf("6. Reverse in linked list 2\n");

printf("7. Concatenate the two lists\n");

printf("8. Display LL 1\n");

printf("9. Display LL 2\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

// Insert in linked list 1

printf("Enter value to insert: ");

scanf("%d", &item);

first\_1 = insert\_end(item, first\_1);

break;

case 2:

// Insert in linked list 2

printf("Enter value to insert: ");

scanf("%d", &item);

first\_2 = insert\_end(item, first\_2);

break;

case 3:

// Sort linked list 1

printf("Sorting LL1\n");

first\_1 = sort(first\_1);

break;

case 4:

// Sort linked list 2

printf("Sorting LL2\n");

first\_2 = sort(first\_2);

break;

case 5:

// Reverse linked list 1

printf("LL1 being reversed\n");

first\_1 = reverse(first\_1);

break;

case 6:

// Reverse linked list 2

printf("LL2 being reversed\n");

first\_2 = reverse(first\_2);

break;

case 7:

// Concatenate the two lists

first\_1 = concatenate(first\_1, first\_2);

first\_2 = NULL; // Optionally clear second list

break;

case 8:

// Display linked list 1

printf("Displaying LL1: ");

display(first\_1);

break;

case 9:

// Display linked list 2

printf("Displaying LL2: ");

display(first\_2);

break;

default:

printf("Invalid choice.\n");

}

}

return 0;

}

Output:

 



