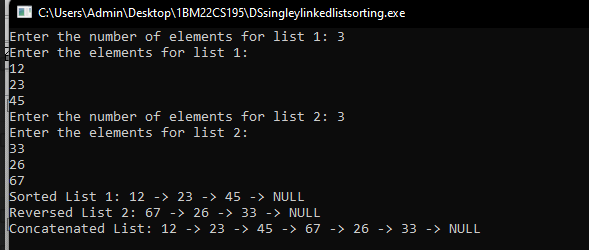
Lab6[28-01-24]

Code:-

#include <stdio.h>  
#include <stdlib.h>  
struct Node {  
    int data;  
    struct Node\* next;  
};  
struct Node\* createNode(int value) {  
    struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));  
    newNode->data = value;  
    newNode->next = NULL;  
    return newNode;  
}  
void insertEnd(struct Node\*\* head, int value) {  
    struct Node\* newNode = createNode(value);  
    if (\*head == NULL) {  
        \*head = newNode;  
    } else {  
        struct Node\* temp = \*head;  
        while (temp->next != NULL) {  
            temp = temp->next;  
        }  
        temp->next = newNode;  
    }  
}  
void display(struct Node\* head) {  
    struct Node\* temp = head;  
    while (temp != NULL) {  
        printf("%d -> ", temp->data);  
        temp = temp->next;  
    }  
    printf("NULL\n");  
}  
void sortLinkedList(struct Node\* head) {  
    int swapped, i;  
    struct Node\* ptr;  
    struct Node\* lptr = NULL;  
    if (head == NULL)  
        return;  
    do {  
        swapped = 0;  
        ptr = head;  
        while (ptr->next != lptr) {  
            if (ptr->data > ptr->next->data) {  
                int temp = ptr->data;  
                ptr->data = ptr->next->data;  
                ptr->next->data = temp;  
                swapped = 1;  
            }  
            ptr = ptr->next;  
        }  
        lptr = ptr;  
    } while (swapped);  
}  
struct Node\* reverseLinkedList(struct Node\* head) {  
    struct Node \*prev = NULL, \*current = head, \*next = NULL;  
    while (current != NULL) {  
        next = current->next;  
        current->next = prev;  
        prev = current;  
        current = next;  
    }  
    return prev;  
}  
void concatenateLinkedLists(struct Node\*\* list1, struct Node\* list2) {  
    if (\*list1 == NULL) {  
        \*list1 = list2;  
    } else {  
        struct Node\* temp = \*list1;  
        while (temp->next != NULL) {  
            temp = temp->next;  
        }  
        temp->next = list2;  
    }  
}  
int main() {  
    struct Node\* list1 = NULL;  
    struct Node\* list2 = NULL;  
    int n, value;  
    printf("Enter the number of elements for list 1: ");  
    scanf("%d", &n);  
    printf("Enter the elements for list 1:\n");  
    for (int i = 0; i < n; i++) {  
        scanf("%d", &value);  
        insertEnd(&list1, value);  
    }  
    printf("Enter the number of elements for list 2: ");  
    scanf("%d", &n);  
    printf("Enter the elements for list 2:\n");  
    for (int i = 0; i < n; i++) {  
        scanf("%d", &value);  
        insertEnd(&list2, value);  
    }  
    sortLinkedList(list1);  
    printf("Sorted List 1: ");  
    display(list1);  
    list2 = reverseLinkedList(list2);  
    printf("Reversed List 2: ");  
    display(list2);  
    concatenateLinkedLists(&list1, list2);  
    printf("Concatenated List: ");  
    display(list1);  
    struct Node\* temp;  
    while (list1 != NULL) {  
        temp = list1;  
        list1 = list1->next;  
        free(temp);  
    }  
    return 0;  
}

Output:-

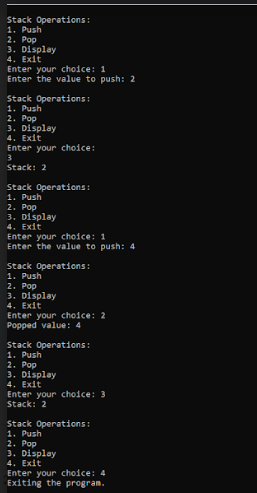


Code:-

Stack

#include <stdio.h>  
#include <stdlib.h>  
struct Node {  
    int data;  
    struct Node\* next;  
};  
struct Node\* createNode(int value) {  
    struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));  
    newNode->data = value;  
    newNode->next = NULL;  
    return newNode;  
}  
void push(struct Node\*\* top, int value) {  
    struct Node\* newNode = createNode(value);  
    newNode->next = \*top;  
    \*top = newNode;  
}  
int pop(struct Node\*\* top) {  
    if (\*top == NULL) {  
        printf("Stack underflow!\n");  
        return -1;  
    }  
    struct Node\* temp = \*top;  
    int poppedValue = temp->data;  
    \*top = temp->next;  
    free(temp);  
    return poppedValue;  
}  
void displayStack(struct Node\* top) {  
    printf("Stack: ");  
    while (top != NULL) {  
        printf("%d ", top->data);  
        top = top->next;  
    }  
    printf("\n");  
}  
int main() {  
    struct Node\* top = NULL;  
    int choice, value;  
    do {  
        printf("\nStack Operations:\n");  
        printf("1. Push\n");  
        printf("2. Pop\n");  
        printf("3. Display\n");  
        printf("4. Exit\n");  
        printf("Enter your choice: ");  
        scanf("%d", &choice);  
        switch (choice) {  
            case 1:  
                printf("Enter the value to push: ");  
                scanf("%d", &value);  
                push(&top, value);  
                break;  
            case 2:  
                value = pop(&top);  
                if (value != -1) {  
                    printf("Popped value: %d\n", value);  
                }  
                break;  
            case 3:  
                displayStack(top);  
                break;  
            case 4:  
                printf("Exiting the program.\n");  
                break;  
            default:  
                printf("Invalid choice! Please enter a valid option.\n");  
        }  
    } while (choice != 4);  
    struct Node\* temp;  
    while (top != NULL) {  
        temp = top;  
        top = top->next;  
        free(temp);  
    }  
    return 0;  
}

Output:-



Code:-

Queues

#include <stdio.h>  
#include <stdlib.h>  
struct Node {  
    int data;  
    struct Node\* next;  
};  
struct Queue {  
    struct Node\* front;  
    struct Node\* rear;  
};  
struct Node\* createNode(int value) {  
    struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));  
    newNode->data = value;  
    newNode->next = NULL;  
    return newNode;  
}  
struct Queue\* createQueue() {  
    struct Queue\* queue = (struct Queue\*)malloc(sizeof(struct Queue));  
    queue->front = queue->rear = NULL;  
    return queue;  
}  
void enqueue(struct Queue\* queue, int value) {  
    struct Node\* newNode = createNode(value);  
    if (queue->rear == NULL) {  
        queue->front = queue->rear = newNode;  
        return;  
    }  
    queue->rear->next = newNode;  
    queue->rear = newNode;  
}  
int dequeue(struct Queue\* queue) {  
    if (queue->front == NULL) {  
        printf("Queue underflow!\n");  
        return -1;  
    }  
    struct Node\* temp = queue->front;  
    int dequeuedValue = temp->data;  
    queue->front = temp->next;  
    if (queue->front == NULL) {  
        queue->rear = NULL;  
    }  
    free(temp);  
    return dequeuedValue;  
}  
void displayQueue(struct Queue\* queue) {  
    struct Node\* temp = queue->front;  
    printf("Queue: ");  
    while (temp != NULL) {  
        printf("%d ", temp->data);  
        temp = temp->next;  
    }  
    printf("\n");  
}  
int main() {  
    struct Queue\* queue = createQueue();  
    int choice, value;  
    do {  
        printf("\nQueue Operations:\n");  
        printf("1. Enqueue\n");  
        printf("2. Dequeue\n");  
        printf("3. Display\n");  
        printf("4. Exit\n");  
        printf("Enter your choice: ");  
        scanf("%d", &choice);  
        switch (choice) {  
            case 1:  
                printf("Enter the value to enqueue: ");  
                scanf("%d", &value);  
                enqueue(queue, value);  
                break;  
            case 2:  
                value = dequeue(queue);  
                if (value != -1) {  
                    printf("Dequeued value: %d\n", value);  
                }  
                break;  
            case 3:  
                displayQueue(queue);  
                break;  
            case 4:  
                printf("Exiting the program.\n");  
                break;  
            default:  
                printf("Invalid choice! Please enter a valid option.\n");  
        }  
    } while (choice != 4);  
    struct Node\* temp;  
    while (queue->front != NULL) {  
        temp = queue->front;  
        queue->front = queue->front->next;  
        free(temp);  
    }  
    free(queue);  
    return 0;  
}

Ouput:-

