// WAP to Implement Single Link List to simulate Stack & Queue Operations.

```
#include <stdio.h>
#include <stdlib.h>

struct node {
   int data;
   struct node *next;
};

// Stack & Queue heads
struct node *top = NULL; // For Stack
struct node *front = NULL; // For Queue
struct node *rear = NULL; // For Queue

// PUSH
void push(int x) {
   struct node *temp = (struct node*)malloc(sizeof(struct node));
   temp->data = x;
   temp->next = top;
   top = temp;
}

// POP
```

```
void pop() {
    if (top == NULL) {
        printf("Stack Underflow!\n");
        return;
    }
    struct node *temp = top;
    printf("Popped element: %d\n", top->data);
    top = top->next;
    free(temp);
}

// DISPLAY STACK
void displayStack() {
    struct node *p = top;
    if (p == NULL) {
        printf("Stack is Empty!\n");
        return;
    }
    printf("Stack: ");
    while (p != NULL) {
        printf("%d ", p->data);
        p = p->next;
    }
    printf("\n");
}
```

```
// ENQUEUE
void enqueue(int x) {
    struct node *temp = (struct node*)malloc(sizeof(struct node));
    temp->data = x;
    temp->next = NULL;

    if (front == NULL) {
        front = rear = temp;
    } else {
        rear->next = temp;
        rear = temp;
    }
}

// DEQUEUE
void dequeue() {
    if (front == NULL) {
        printf("Queue Underflow!\n");
        return;
    }
    struct node *temp = front;
    printf("Dequeued element: %d\n", front->data);
    front = front->next;
```

```
    if (front == NULL)
        rear = NULL;

    free(temp);
}

// DISPLAY QUEUE
void displayQueue() {
    struct node *p = front;
    if (p == NULL) {
        printf("Queue is Empty!\n");
        return;
    }
    printf("Queue: ");
    while (p != NULL) {
        printf("%d ", p->data);
        p = p->next;
    }
    printf("\n");
}

int main() {
    int choice, item;

    while (1) {
```

```
printf("\n===== MENU =====\n");
printf("1. Push (Stack)\n");
printf("2. Pop (Stack)\n");
printf("3. Display Stack\n");
printf("4. Enqueue (Queue)\n");
printf("5. Dequeue (Queue)\n");
printf("6. Display Queue\n");
printf("7. Exit\n");
printf("Enter choice: ");
scanf("%d", &choice);

switch (choice) {
case 1:
   printf("Enter element to push: ");
   scanf("%d", &item);
   push(item);
   break;

case 2:
   pop();
   break;

case 3:
   displayStack();
   break;
```

```
        case 4:
            printf("Enter element to enqueue: ");
            scanf("%d", &item);
            enqueue(item);
            break;

        case 5:
            dequeue();
            break;

        case 6:
            displayQueue();
            break;

        case 7:
            exit(0);

        default:
            printf("Invalid choice!\n");
        }
    }
    return 0;
}
```

OUTPUT:-


"C:\Users\88nin\OneDrive\Do
===== MENU =====
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter choice: 1
Enter element to push: 45
===== MENU =====
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter choice: 1
Enter element to push: 55
===== MENU =====
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter choice: 2
Popped element: 55
===== MENU =====
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit


"C:\Users\88nin\OneDrive\Do
===== MENU =====
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter choice: 55
Invalid choice!
===== MENU =====
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter choice: 3
Stack: 45
===== MENU =====
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter choice: 4
Enter element to enqueue: 70
===== MENU =====
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit


"C:\Users\88nin\OneDrive\Do
Enter choice: 4
Enter element to enqueue: 70
===== MENU =====
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter choice: 5
Dequeued element: 70
===== MENU =====
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter choice: 6
Queue is Empty!
===== MENU =====
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter choice: 7
Process returned 0 (0x0) execution time : 66.281 s
Press any key to continue.