LAB PROGRAM 6b

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

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

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

struct node *top = NULL;
struct node *front = NULL;
struct node *rear = NULL;

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

void pop() {
    struct node *temp;
    if (top == NULL) {
        printf("Stack Underflow\n");
        return;
```

```
    }
    temp = top;
    top = top->next;
    free(temp);
}

void displayStack() {
    struct node *temp = top;
    printf("Stack: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

void enqueue(int x) {
    struct node *newnode = (struct node *)malloc(sizeof(struct node));
    newnode->data = x;
    newnode->next = NULL;

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

```
void dequeue() {
    struct node *temp;
    if (front == NULL) {
        printf("Queue Underflow\n");
        return;
    }
    temp = front;
    front = front->next;
    if (front == NULL)
        rear = NULL;
    free(temp);
}

void displayQueue() {
    struct node *temp = front;
    printf("Queue: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    int choice, x;

    while (1) {
        printf("\n1.Push(Stack)\n2.Pop(Stack)\n3.Display
Stack\n4.Enqueue(Queue)\n5.Dequeue(Queue)\n6.Display Queue\n7.Exit\n");
```

```
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            scanf("%d", &x);
            push(x);
            break;
        case 2:
            pop();
            break;
        case 3:
            displayStack();
            break;
        case 4:
            scanf("%d", &x);
            enqueue(x);
            break;
        case 5:
            dequeue();
            break;
        case 6:
            displayQueue();
            break;
        case 7:
            exit(0);
        default:
            printf("Invalid choice\n");
    }
}
```

}

OUTPUT:

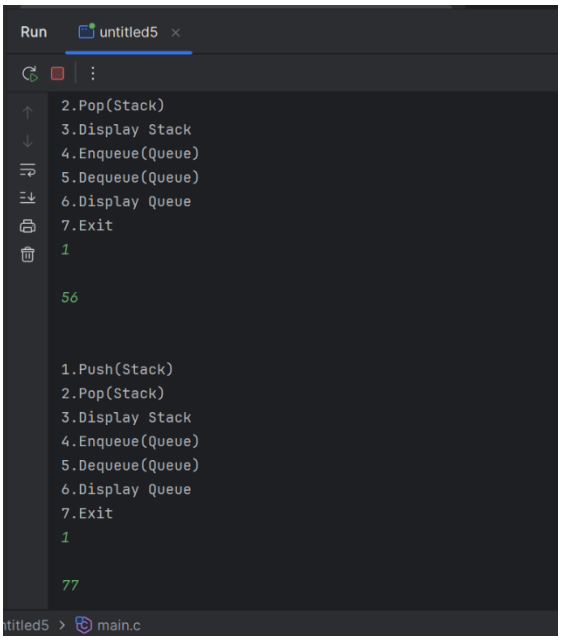
```
1.Push(Stack)
2.Pop(Stack)
3.Display Stack
4.Enqueue(Queue)
5.Dequeue(Queue)
6.Display Queue
7.Exit
1

23
```

```
1.Push(Stack)
2.Pop(Stack)
3.Display Stack
4.Enqueue(Queue)
5.Dequeue(Queue)
6.Display Queue
7.Exit
4

22


1.Push(Stack)
2.Pop(Stack)
3.Display Stack
4.Enqueue(Queue)
5.Dequeue(Queue)
6.Display Queue
7.Exit
4
```




```
1.Push(Stack)
2.Pop(Stack)
3.Display Stack
4.Enqueue(Queue)
5.Dequeue(Queue)
6.Display Queue
7.Exit
5


1.Push(Stack)
2.Pop(Stack)
3.Display Stack
4.Enqueue(Queue)
5.Dequeue(Queue)
6.Display Queue
7.Exit
6

Queue: 56
```

```
1.Push(Stack)
2.Pop(Stack)
3.Display Stack
4.Enqueue(Queue)
5.Dequeue(Queue)
6.Display Queue
7.Exit
2


1.Push(Stack)
2.Pop(Stack)
3.Display Stack
4.Enqueue(Queue)
5.Dequeue(Queue)
6.Display Queue
7.Exit
3

Stack: 56 23
```