6b. WAP to Implement Single Link List to simulate Stack & Queue Operations.

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

// Node structure
struct Node {
    int data;
    struct Node *next;
};

// Global pointers
struct Node *top = NULL;      // For Stack
struct Node *front = NULL;    // For Queue
struct Node *rear = NULL;

// ---------------- STACK OPERATIONS ----------------

// Push operation (Stack)
void push(int value) {
    struct Node *newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = top;
    top = newNode;
    printf("Pushed %d into stack\n", value);
}

// Pop operation (Stack)
void pop() {
    if (top == NULL) {
        printf("Stack Underflow\n");
        return;
    }
```

```
    struct Node *temp = top;
    printf("Popped %d from stack\n", temp->data);
    top = top->next;
    free(temp);
}

// Display Stack
void displayStack() {
    struct Node *temp = top;
    if (temp == NULL) {
        printf("Stack is empty\n");
        return;
    }
    printf("Stack: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

// ---------- QUEUE OPERATIONS ----------

// Enqueue operation (Queue)
void enqueue(int value) {
    struct Node *newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = NULL;

    if (rear == NULL) {
        front = rear = newNode;
    } else {
```

```
        rear->next = newNode;
        rear = newNode;
    }
    printf("Enqueued %d into queue\n", value);
}

// Dequeue operation (Queue)
void dequeue() {
    if (front == NULL) {
        printf("Queue Underflow\n");
        return;
    }
    struct Node *temp = front;
    printf("Dequeued %d from queue\n", temp->data);
    front = front->next;
    if (front == NULL)
        rear = NULL;
    free(temp);
}

// Display Queue
void displayQueue() {
    struct Node *temp = front;
    if (temp == NULL) {
        printf("Queue is empty\n");
        return;
    }
    printf("Queue: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
```

```
    printf("\n");
}

// ---------------- MAIN FUNCTION ----------------

int main() {
    int choice, value;

    while (1) {
        printf("\nMenu:\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 your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value: ");
                scanf("%d", &value);
                push(value);
                break;
            case 2:
                pop();
                break;
            case 3:
                displayStack();
                break;
```

```
            case 4:
                printf("Enter value: ");
                scanf("%d", &value);
                enqueue(value);
                break;
            case 5:
                dequeue();
                break;
            case 6:
                displayQueue();
                break;
            case 7:
                return 0;
            default:
                printf("Invalid choice\n");
        }
    }
}
```

```
Menu:
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 1
Enter value: 10
Pushed 10 into stack

Menu:
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 1
Enter value: 20
Pushed 20 into stack

Menu:
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 1
Enter value: 30
Pushed 30 into stack

Menu:
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 2
Popped 30 from stack

Menu:
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 3
Stack: 20 10
```

```
Menu:
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 4
Enter value: 30
Enqueued 30 into queue

Menu:
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 5
Dequeued 30 from queue

Menu:
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your 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 your choice: 7

Process returned 0 (0x0)   execution time : 45.946 s
Press any key to continue.
```