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

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

// Definition of node
struct node {
   int data;
   struct node *next;
};

// Top pointer for Stack
struct node *top = NULL;

// Front and Rear pointers for Queue
struct node *front = NULL, *rear = NULL;

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

// Push operation (Stack)
void push() {
   int value;
   struct node *newnode = (struct node *)malloc(sizeof(struct node));

   printf("Enter value to push: ");
   scanf("%d", &value);
```

```
    newnode->data = value;
    newnode->next = top;
    top = newnode;

    printf("%d pushed into stack.\n", value);
}

// Pop operation (Stack)
void pop() {
    if (top == NULL) {
        printf("Stack is empty. Cannot pop.\n");
        return;
    }

    struct node *temp = top;
    printf("%d popped from stack.\n", temp->data);
    top = top->next;
    free(temp);
}

// Display stack
void display_stack() {
    struct node *temp = top;

    if (top == NULL) {
        printf("Stack is empty.\n");
        return;
    }
```

```
    printf("Stack elements:\n");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

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

// Enqueue operation (Queue)
void enqueue() {
    int value;
    struct node *newnode = (struct node *)malloc(sizeof(struct node));

    printf("Enter value to enqueue: ");
    scanf("%d", &value);

    newnode->data = value;
    newnode->next = NULL;

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

```
    printf("%d enqueued into queue.\n", value);
}

// Dequeue operation (Queue)
void dequeue() {
    if (front == NULL) {
        printf("Queue is empty. Cannot dequeue.\n");
        return;
    }

    struct node *temp = front;
    printf("%d dequeued from queue.\n", temp->data);
    front = front->next;

    if (front == NULL)
        rear = NULL;

    free(temp);
}

// Display queue
void display_queue() {
    struct node *temp = front;

    if (front == NULL) {
        printf("Queue is empty.\n");
        return;
```

```
    }

    printf("Queue elements:\n");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

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

int main() {
    int choice;

    do {
        printf("\n--- Stack & Queue Using Linked List ---\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: push(); break;
            case 2: pop(); break;
            case 3: display_stack(); break;
            case 4: enqueue(); break;
            case 5: dequeue(); break;
            case 6: display_queue(); break;
            case 7: printf("Exiting program.\n"); break;
            default: printf("Invalid choice!\n");
        }
    } while (choice != 7);

    return 0;
}
```

OUTPUT:

```
--- Stack & Queue Using Linked List ---
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 to push: 10
10 pushed into stack.

--- Stack & Queue Using Linked List ---
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 to push: 20
20 pushed into stack.

--- Stack & Queue Using Linked List ---
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 to push: 30
30 pushed into stack.

--- Stack & Queue Using Linked List ---
1. Push (Stack)
```

```
--- Stack & Queue Using Linked
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 to push: 40
40 pushed into stack.

--- Stack & Queue Using Linked
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 to push: 50
50 pushed into stack.

--- Stack & Queue Using Linked
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 2
50 popped from stack.

--- Stack & Queue Using Linked
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
```

```
6. Display Queue
7. Exit
Enter your choice: 3
Stack elements:
40 -> 30 -> 20 -> 10 -> NULL

--- Stack & Queue Using Linked List ---
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 to enqueue: 5
5 enqueued into queue.

--- Stack & Queue Using Linked List ---
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 elements:
5 -> NULL

--- Stack & Queue Using Linked List ---
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 7
Exiting program.
```