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

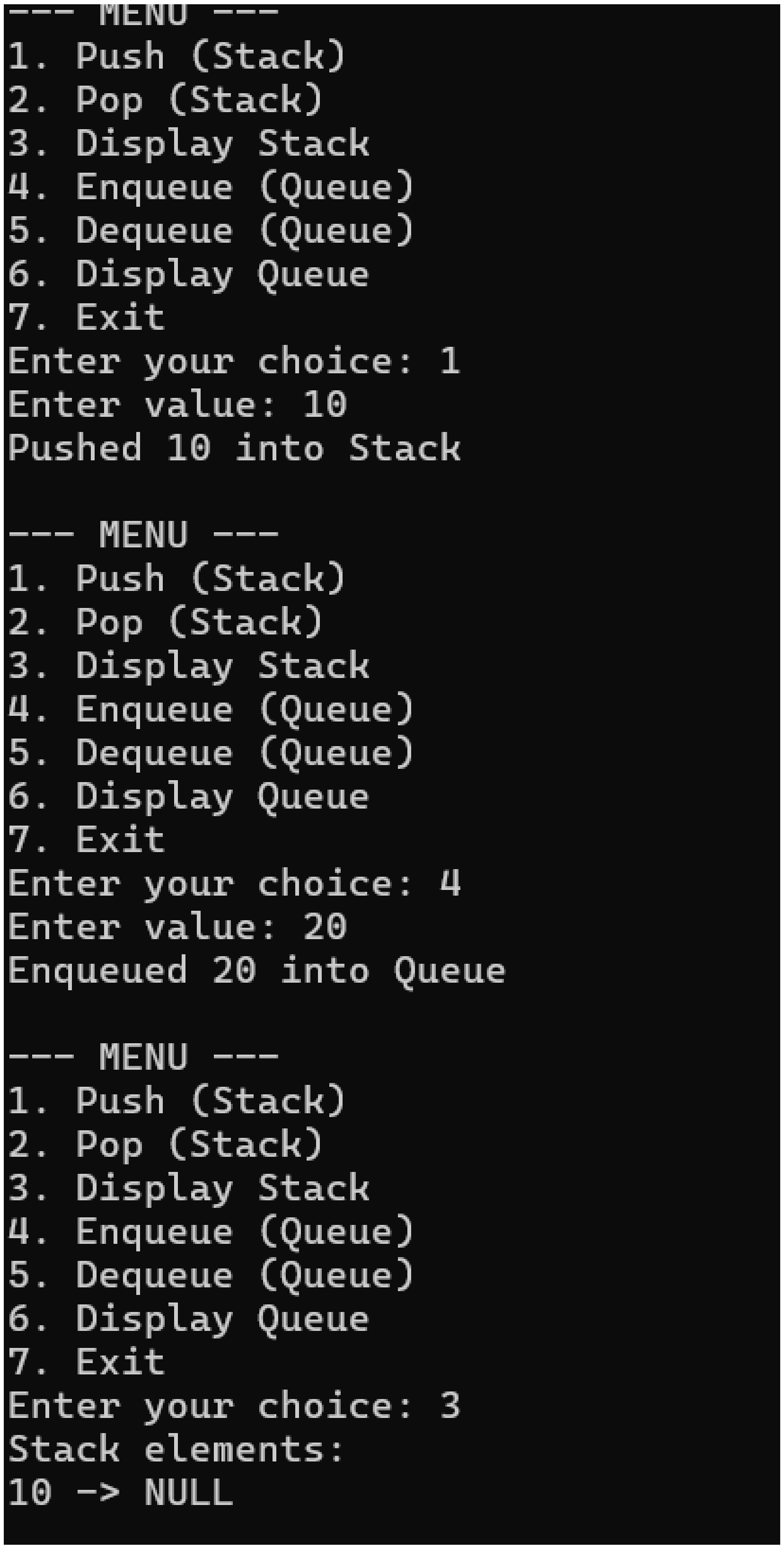
```
#include <stdio.h>

#include <stdlib.h>


/* Node structure */

struct node {

   int data;

   struct node *next;

};


/* ---------- STACK USING LINKED LIST ---------- */

struct node *top = NULL;


/* Push operation */

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 */

void pop() {

   if (top == NULL) {

      printf("Stack is Empty\n");

      return;

   }
```

```
    struct node *temp = top;
    printf("Popped element: %d\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 elements:\n");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

/* ---------- QUEUE USING LINKED LIST ---------- */
struct node *front = NULL, *rear = NULL;

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

    if (front == NULL)
        rear = NULL;

    free(temp);
}

/* Display Queue */
```

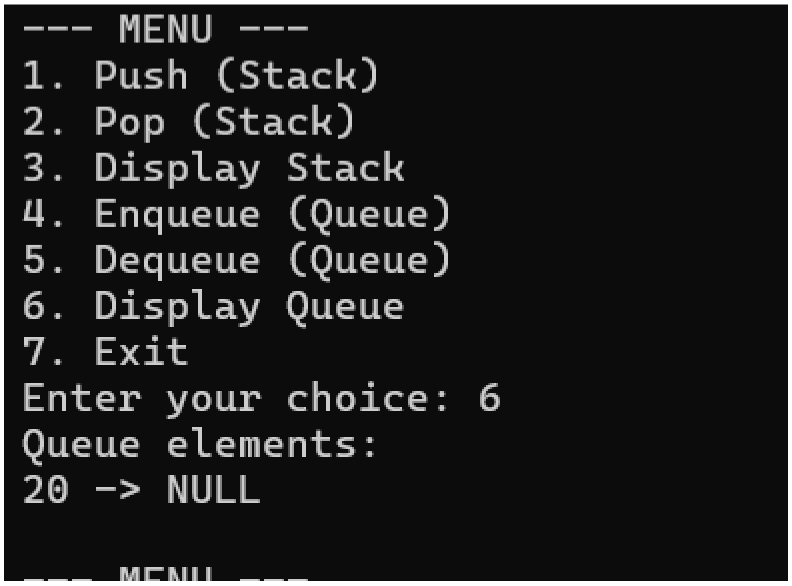
```
void displayQueue() {
    struct node *temp = front;
    if (temp == NULL) {
        printf("Queue is Empty\n");
        return;
    }
    printf("Queue elements:\n");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

/* ---------- MAIN FUNCTION ---------- */
int main() {
    int choice, value;

    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 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:
            exit(0);

        default:
            printf("Invalid Choice\n");
        }
    Return o;
}
```



```
--- 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: 4
Enter value: 20
Enqueued 20 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: 3
Stack elements:
10 -> NULL
```



```
--- 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 elements:
20 -> NULL

--- MENU ---
```