

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

Code for stack:

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

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

```
struct node
```

```
{
```

```
    int data;
```

```
    struct node *next;
```

```
};
```

```
struct node *head = NULL;
```

```
void push(int val)
```

```
{
```

```
    struct node *newNode = malloc(sizeof(struct node));
```

```
    newNode->data = val;
```

```
    newNode->next = head;
```

```
    head = newNode;
```

```
}
```

```
void pop()
```

```
{
```

```
    struct node *temp;
```

```
    if(head == NULL)
```

```
        printf("Stack is Empty\n");
```

```
    else
```

```
    {
```

```
        printf("Popped element = %d\n", head->data);
```

```
        temp = head;
```

```
        head = head->next;
```

```
        free(temp);
```

```
    }
```

```
}
```

```
void printList()
```

```
{
```

```
    struct node *temp = head;
```

```
    while(temp != NULL)
```

```
    {
```

```
        printf("%d->", temp->data);
```

```

        temp = temp->next;
    }
    printf("NULL\n");
}

int main()
{
    int data, ch;
    printf("Menu:\n 1. Push\n 2. Pop\n 3. Display\n 4. Exit");
    printf("\nEnter choice: ");
    scanf("%d",&ch);
    while(ch!=4){
        switch(ch){
            case 1:
                printf("Enter data to be pushed: ");
                scanf("%d",&data);
                push(data);
                break;
            case 2:
                pop();
                break;
            case 3:
                printList();
                break;
            case 4:
                exit(0);
        }
        printf("\nEnter choice: ");
        scanf("%d",&ch);
    }
    return 0;
}

```

Output:

```
Menu:
 1. Push
 2. Pop
 3. Display
 4. Exit
Enter choice: 1
Enter data to be pushed: 1

Enter choice: 1
Enter data to be pushed: 2

Enter choice: 1
Enter data to be pushed: 3

Enter choice: 3
3->2->1->NULL

Enter choice: 2
Popped element = 3

Enter choice: 3
2->1->NULL

Enter choice: 2
Popped element = 2

Enter choice: 2
Popped element = 1

Enter choice: 2
Stack is Empty

Enter choice: 4

Process returned 0 (0x0)   execution time : 29.365 s
Press any key to continue.
```

Code for queues:

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

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

```
struct node *front = NULL, *rear = NULL;
```

```

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

    //if it is the first node
    if(front == NULL && rear == NULL)
        //make both front and rear points to the new node
        front = rear = newNode;
    else
    {
        //add newnode in rear->next
        rear->next = newNode;

        //make the new node as the rear node
        rear = newNode;
    }
}

void dequeue()
{
    //used to free the first node after dequeue
    struct node *temp;

    if(front == NULL)
        printf("Queue is Empty. Unable to perform dequeue\n");
    else
    {
        //take backup
        temp = front;

        //make the front node points to the next node
        //logically removing the front element
        front = front->next;

        //if front == NULL, set rear = NULL
        if(front == NULL)
            rear = NULL;

        //free the first node
        free(temp);
    }
}

```

```
}
```

```
void printList()
```

```
{
```

```
    struct node *temp = front;
```

```
    while(temp)
```

```
    {
```

```
        printf("%d->",temp->data);
```

```
        temp = temp->next;
```

```
    }
```

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

```
}
```

```
int main()
```

```
{
```

```
    int data, ch;
```

```
    printf("Menu:\n 1. Enqueue\n 2. Dequeue\n 3. Display\n 4. Exit");
```

```
    printf("\nEnter choice: ");
```

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

```
    while(ch!=4){
```

```
        switch(ch){
```

```
            case 1:
```

```
                printf("Enter data to be pushed: ");
```

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

```
                enqueue(data);
```

```
                break;
```

```
            case 2:
```

```
                dequeue();
```

```
                break;
```

```
            case 3:
```

```
                printList();
```

```
                break;
```

```
            case 4:
```

```
                exit(0);
```

```
        }
```

```
        printf("\nEnter choice: ");
```

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

```
    }
```

```
    return 0;
```

```
}
```

Output:

```
Menu:
 1. Enqueue
 2. Dequeue
 3. Display
 4. Exit
Enter choice: 1
Enter data to be pushed: 1

Enter choice: 1
Enter data to be pushed: 2

Enter choice: 1
Enter data to be pushed: 3

Enter choice: 3
1->2->3->NULL

Enter choice: 2

Enter choice:
3
2->3->NULL

Enter choice: 2

Enter choice: 3
3->NULL

Enter choice: 2

Enter choice: 2
Queue is Empty. Unable to perform dequeue

Enter choice: 4

Process returned 0 (0x0)   execution time : 30.701 s
Press any key to continue.
```