Lab program 6b:-

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

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

/* Structure of Node */
struct node
{
   int data;
   struct node *next;
};

struct node *head = NULL;

/* Function Prototypes */
void push();       // Stack Push
void pop();        // Stack Pop
void enqueue();    // Queue Enqueue
void dequeue();    // Queue Dequeue
void display();

/* Main Function */
int main()
{
   int choice;

   while (1)
   {
      printf("\n----- MENU -----");
```

```
printf("\n1. Stack Push");
printf("\n2. Stack Pop");
printf("\n3. Queue Enqueue");
printf("\n4. Queue Dequeue");
printf("\n5. Display");
printf("\n6. Exit");
printf("\nEnter your choice: ");
scanf("%d", &choice);

switch (choice)
{
case 1:
   push();
   break;
case 2:
   pop();
   break;
case 3:
   enqueue();
   break;
case 4:
   dequeue();
   break;
case 5:
   display();
   break;
case 6:
   exit(0);
default:
```

```
            printf("\nInvalid choice!");
        }
    }
    return 0;
}

/* Stack Push (Insert at Beginning) */
void push()
{
    struct node *newnode;
    newnode = (struct node *)malloc(sizeof(struct node));

    if (newnode == NULL)
    {
        printf("\nMemory Overflow!");
        return;
    }

    printf("\nEnter data: ");
    scanf("%d", &newnode->data);

    newnode->next = head;
    head = newnode;

    printf("Push successful.");
}

/* Stack Pop (Delete from Beginning) */
void pop()
```

```
{
   struct node *temp;

   if (head == NULL)
   {
      printf("\nStack Underflow!");
      return;
   }

   temp = head;
   printf("\nPopped element: %d", temp->data);
   head = head->next;
   free(temp);
}

/* Queue Enqueue (Insert at End) */
void enqueue()
{
   struct node *newnode, *temp;
   newnode = (struct node *)malloc(sizeof(struct node));

   if (newnode == NULL)
   {
      printf("\nMemory Overflow!");
      return;
   }

   printf("\nEnter data: ");
   scanf("%d", &newnode->data);
```

```
    newnode->next = NULL;

    if (head == NULL)
    {
        head = newnode;
    }
    else
    {
        temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newnode;
    }

    printf("Enqueue successful.");
}

/* Queue Dequeue (Delete from Beginning) */
void dequeue()
{
    struct node *temp;

    if (head == NULL)
    {
        printf("\nQueue Underflow!");
        return;
    }

    temp = head;
```

```
    printf("\nDequeued element: %d", temp->data);
    head = head->next;
    free(temp);
}

/* Display Linked List */
void display()
{
    struct node *temp;

    if (head == NULL)
    {
        printf("\nList is empty.");
        return;
    }

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

```
----- MENU -----
1. Stack Push
2. Stack Pop
3. Queue Enqueue
4. Queue Dequeue
5. Display
6. Exit
Enter your choice: 1

Enter data: 10
Push successful.
----- MENU -----
1. Stack Push
2. Stack Pop
3. Queue Enqueue
4. Queue Dequeue
5. Display
6. Exit
Enter your choice: 1

Enter data: 20
Push successful.
----- MENU -----
1. Stack Push
2. Stack Pop
3. Queue Enqueue
4. Queue Dequeue
5. Display
6. Exit
Enter your choice: 1

Enter data: 30
```

```
----- MENU -----
1. Stack Push
2. Stack Pop
3. Queue Enqueue
4. Queue Dequeue
5. Display
6. Exit
Enter your choice: 5

List elements: 30 -> 20 -> 10 -> NULL
----- MENU -----
1. Stack Push
2. Stack Pop
3. Queue Enqueue
4. Queue Dequeue
5. Display
6. Exit
Enter your choice: 2

Popped element: 30
----- MENU -----
1. Stack Push
2. Stack Pop
3. Queue Enqueue
4. Queue Dequeue
5. Display
6. Exit
Enter your choice: 5

List elements: 20 -> 10 -> NULL
----- MENU -----
1. Stack Push
2. Stack Pop
3. Queue Enqueue
4. Queue Dequeue
5. Display
6. Exit
Enter your choice: 3

Enter data: 20
```

```
Enqueue successful.
----- MENU -----
1. Stack Push
2. Stack Pop
3. Queue Enqueue
4. Queue Dequeue
5. Display
6. Exit
Enter your choice: 3

Enter data: 40
Enqueue successful.
----- MENU -----
1. Stack Push
2. Stack Pop
3. Queue Enqueue
4. Queue Dequeue
5. Display
6. Exit
Enter your choice: 3

Enter data: 60
Enqueue successful.
----- MENU -----
1. Stack Push
2. Stack Pop
3. Queue Enqueue
4. Queue Dequeue
5. Display
6. Exit
Enter your choice: 5
```

```
List elements: 20 -> 10 -> 20 -> 40 -> 60 -> NULL
----- MENU -----
1. Stack Push
2. Stack Pop
3. Queue Enqueue
4. Queue Dequeue
5. Display
6. Exit
Enter your choice: 4

Dequeued element: 20
----- MENU -----
1. Stack Push
2. Stack Pop
3. Queue Enqueue
4. Queue Dequeue
5. Display
6. Exit
Enter your choice: 5

List elements: 10 -> 20 -> 40 -> 60 -> NULL
----- MENU -----
1. Stack Push
2. Stack Pop
3. Queue Enqueue
4. Queue Dequeue
5. Display
6. Exit
Enter your choice: 6

Process returned 0 (0x0)   execution time : 61.044 s
Press any key to continue.
```