

Lab - 29/01/24

Implement Stacks & Queues using Linked List

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

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

```
#include <conio.h>
```

```
#include <malloc.h>
```

```
struct stack
```

```
{
```

```
    int data;
```

```
    struct stack *next;
```

```
};
```

```
struct stack *top = NULL;
```

```
struct stack *push(struct stack *, int);
```

```
struct stack *display(struct stack *);
```

```
struct stack *pop(struct stack *);
```

```
int peek(struct stack *);
```

```
int main(int argc, char *argv[])
```

```
{
```

```
    top = push(top,100);
```

```
    top = push(top,200);
```

```
    top = display(top);
```

```
    top = pop(top);
```

```
    top = display(top);  
}
```

```
struct stack *push(struct stack *top, int val)  
{  
    struct stack *ptr;  
    ptr = (struct stack *)malloc(sizeof(struct stack));  
    ptr->data = val;  
    if (top == NULL)  
    {  
        ptr->next = NULL;  
        top = ptr;  
    }  
    else  
    {  
        ptr->next = top;  
        top = ptr;  
    }  
    printf("%d pushed to stack\n",val);  
    return top;  
}
```

```
struct stack *display(struct stack *top)  
{  
    struct stack *ptr;  
    ptr = top;
```

```
if (top == NULL)

    printf("\n STACK IS EMPTY");

else

{

    while (ptr != NULL)

    {

        printf("\n %d", ptr->data);

        ptr = ptr->next;

    }

}

return top;

}

struct stack *pop(struct stack *top)

{

    struct stack *ptr;

    ptr = top;

    if (top == NULL)

        printf("\n STACK UNDERFLOW");

    else

    {

        top = top->next;

        printf("\n The value being deleted is: %d", ptr->data);

        free(ptr);

    }

    return top;

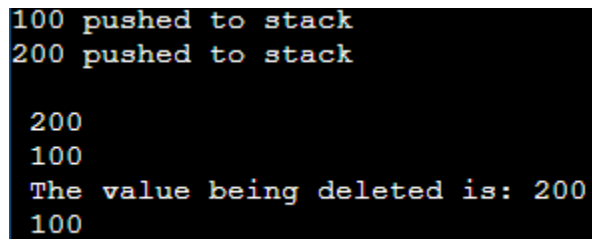
}
```

```

}

int peek(struct stack *top)
{
    if (top == NULL)
        return -1;
    else
        return top->data;
}

```



```

100 pushed to stack
200 pushed to stack

200
100
The value being deleted is: 200
100

```

```

#include <stdio.h>

#include <conio.h>

#include <malloc.h>

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

struct queue
{
    struct node *front;
    struct node *rear;
};

struct queue *q;

```

```

void create_queue(struct queue *);

struct queue *insert(struct queue *, int);

struct queue *delete_element(struct queue *);

struct queue *display(struct queue *);

int peek(struct queue *);

int main()
{
    int val, option;

    create_queue(q);

    q = insert(q,100);

    q = insert(q,200);

    q = display(q);

    q = delete_element(q);

    q = display(q);

    return 0;
}

void create_queue(struct queue *q)
{
    q->rear = NULL;

    q->front = NULL;
}

struct queue *insert(struct queue *q, int val)
{
    struct node *ptr;

    ptr = (struct node *)malloc(sizeof(struct node));

    ptr->data = val;

    if (q->front == NULL)
    {

```

```

    q->front = ptr;
    q->rear = ptr;
    q->front->next = q->rear->next = NULL;
}
else
{
    q->rear->next = ptr;
    q->rear = ptr;
    q->rear->next = NULL;
}
return q;
}

struct queue *display(struct queue *q)
{
    struct node *ptr;
    ptr = q->front;
    if (ptr == NULL)
        printf("\n QUEUE IS EMPTY");
    else
    {
        printf("\n");
        while (ptr != q->rear)
        {
            printf("%d\t", ptr->data);
            ptr = ptr->next;
        }
        printf("%d\t", ptr->data);
    }
}

```

```

    return q;
}

struct queue *delete_element(struct queue *q)
{
    struct node *ptr;

    ptr = q->front;

    if (q->front == NULL)
        printf("\n UNDERFLOW");
    else
    {
        q->front = q->front->next;

        printf("\n The value being deleted is : %d", ptr->data);

        free(ptr);
    }

    return q;
}

int peek(struct queue *q)
{
    if (q->front == NULL)
    {
        printf("\n QUEUE IS EMPTY");

        return -1;
    }

    else

        return q->front->data;
}

```

```
100 pushed to queue
200 pushed to queue
100
200The value being deleted is 100
200
```