Data Structure Lab Test.

Courses: Data Structure with C.

Course Code: 19IS3PCDSC

Date - 15/01/2021

Section - 3D.

USN - 1BM19IS198

Name - Jay Bhattarai

Signature -

Program Description.

Implement circular/linear queue using array.

Program

```
#include <stdio.h>
#define MAXSIZE 5
int cArray[MAXSIZE]
int front = -1;
int rear = -1;
void enqueue (int item)
{
    if ((front == 0 && rear == MAXSIZE-1) || (front == rear+1))
    {
        printf("Queue Overflow\n");
        return;
    }
    if (front == -1)
    {
        front = 0;
        rear = 0;
    }
```

```
else
{
    if (rear == MAXSIZE -1)
        rear = 0;
    else
        rear = rear + 1;

}.
cArray [rear] = item;

}.
void dequeue ()
{
    if (front == -1)
    {
        printf ( "Queue Underflow \n");
        return;
    }
    printf ("Enter Element deleted from queue is: %d ", cArray [front]);
    if (front == rear)
    {
        front = -1;
        rear = -1;

    }
    else
    {
        if (front == MAXSIZE -1)
            front = 0;
        else
            front = front + 1;
    }
}.
```

```
void display()
{
    int frontPos = front, rearPos = rear;
    if (front == -1)
    {
        printf("Queue is empty \n");
        return;
    }.
    printf("Queue elements : \n");
    if (frontPos ≤ rearPos)
        while (front pos ≤ rearPos)
        {
            printf(" %d \n", cArray[frontPos]);
            frontPos++;
        }.
    else.
    {
        while (frontPos ≤ MAXSIZE -1)
        {
            printf("%d", cArray [frontPos]);
            frontPos++;
        }.
        frontPos = 0;
        while (frontPos ≤ rearPos)
        {
            printf("%d", cArray [frontpos]);
            frontPos++;
        }.
    }printf("\n");
}.
```

```
int main()
{
    int choice, item;
    do
    {
        printf("\n 1: Insert \n 2: Delete \n 3: Display \n 4: Quit \n");
        printf("Enter your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                printf("Input the element for insertion ");
                scanf("%d", &item);
                enqueue (item);
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display ();
                break;
            case 4:
                break;
            default :
                printf("Enter valid choice \n");
        }.
    } while (choice != 4);
    return 0;
}.
```

USN- 1BM19IS198

Name - Jay Bhattarai.
Signature -

Expected Output.

~~Enter your choice.~~
1. Insert.
2. Delete
3. Display.
4. Quit.

Enter your choice : 1.

Input the element for insertion in queue : 3.

1. Insert
2. Delete.
3. Display
4. Quit.

Enter your choice : 1.

Input the element for insertion in queue : 6.

1. Insert.
2. Delete
3. Display
4. Quit.

Enter your choice : 3.

3
6.

Application Program

Implement a queue using a two stacks.

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

struct node.
{
    int data;
    struct node *next;
}.
void push(struct node **top, int data);
int pop(struct node ** top);

struct queue.
{
    struct node *stack1;
    struct node *stack2;
}
void enqueue(struct queue *q, int x).
{
    push(&q->stack1, x);
}
void dequeue(struct queue *q)
{
    int x;
    if(q->stack1 == NULL && q->stack2 == NULL)
    {
        print("queue is empty");
        return;
    if(q->stack2 == NULL)
    {
        while (q->stack1 != NULL)
        {
```

```
            x = pop(&q->stack1);
            push(&q->stack2, x);
        }
    }.
    x = pop(&q->stack2);
    printf("%d \n", x);
}.
void push(struct node **top, int data)
{
    struct node *newnode = (struct node *)malloc(sizeof
                                (struct node));
    if(newnode == NULL)
    {
        printf("Stack Overflow \n");
        return;
    }.
    ~~retu~~ newnode -> data = data;
    newnode -> next = (*top);
    (*top) = newnode;
}.
int pop( struct node **top)
{
    int del;
    struct node *t;
    if (*top = NULL)
    {
        printf(" Stack Underflow \n");
    }.
```

```
else
{
        t = *top;
        del = t->data;
        *top = t->next;
        free(t);
        return del;
    }.

}.
void display(struct node *top1, struct node *top2)
{
    while (top1 != NULL)
    {
        printf("%d \n", top1->data);
        top1 = top1->next;
    }.
    while (top2 != NULL)
    {
        printf("%d\n", top2->data);
        top2 = top2->next;
    }.

}.
int main()
{
    struct queue *q = (struct queue *) malloc(sizeof(struct queue));
    int f=0, a;
    char ch='y';
    q->stack1 = NULL;
```

```
q -> stack2 = NULL;
while(ch == "Y" || ch == "y").
{
    printf("Enter ur choice \n 1. add to queue \n 2. remove
            from queue \n 3. display \n 4. Exit \n");
    scanf("%d", &f);
    switch (f)
    {
        case 1:
            printf("Enter the element to be added to
                    queue \n");
            scanf("%d", &a);
            enqueue (q, a).
            break;
        case 2:
            dequeue (q);
            break;
        case 3:
            display (q -> stack 1, q -> stack 2);
            break;
        case 4:
            exit(1);
            break;
        default:
            printf("invalid \n");
            break;
    }.
  }.
}.
```

Expected Output

Enter your choice.
1. Add to queue
2. Remove from queue
3. display
4. exit.

1.
Enter the element to be added to queue.
36.

1.
Enter element to be added to queue.
44.

1
Enter element to be added to queue
55

3
55 44 36.

2.

3
55 44.