VISHWAKARMA INSTITUTE OF TECHNOLOGY
DATA STRUCTURE ASSIGNMENT

NAME	Arpit Sudhir Vidhale
ROLL NO.	60
DIVISION	CS-D
BATCH	B3
PRN NO.	12111229

ASSIGNMENT 4

Question :

Write a Program to implement double ended queue where user can add and remove the elements from both front and rear of the queue

Code :

```
#include<stdio.h>
#define MAX 5

int deque[MAX];
int left = -1, right = -1;

void insert_right (void);
void insert_left (void);
void delete_right (void);
void delete_left (void);
void display (void);

int main ()
{
  int choice;
  do
```

```
    {
      printf ("1.Insert at right ");
      printf ("\t2.Insert at left ");
      printf ("\t3.Delete from right ");
      printf ("\n4.Delete from left ");
      printf ("\t5.Display the queue
      "); printf ("\t6.Exit the session");
      printf ("\nEnter your choice ");
      scanf ("%d", &choice);
      switch (choice)
          {
          case 1:
            insert_right ();
            break;
          case 2:
            insert_left ();
            break;
          case 3:
            delete_right ();
            break;
          case 4:
            delete_left ();
            break;
          case 5:
            display ();
            break;
          }
    }
  while (choice != 6);
  return 0;
}
```

```
void insert_right ()
{
  int val;
  printf ("\nEnter the value to be added ");
  scanf ("%d", &val);
  if ((left == 0 && right == MAX - 1) || (left == right +
    1)) {
      printf ("\nOVERFLOW");
    }
  if (left == -1)
    {
      left = 0;
      right = 0;
    }
  else
    {
      if (right == MAX - 1)
            right = 0;
      else
            right = right + 1;
    }
  deque[right] = val;
}

void insert_left ()
{
  int val;
  printf ("\nEnter the value to be added ");
  scanf ("%d", &val);
  if ((left == 0 && right == MAX - 1) || (left == right +
```

```
    1)) {
      printf ("\nOVERFLOW");
    }
  if (left == -1)
    {
      left = 0;
      right = 0;
    }
  else
    {
      if (left == 0)
           left = MAX - 1;
      else
           left = left - 1;
    }
  deque[left] = val;
}

void delete_right ()
{
  if (left == -1)
    {
      printf ("\nUNDERFLOW");
      return;
    }
  printf ("\nThe deleted element is %d\n",
  deque[right]); if (left == right)
    {
      left = -1;
      right = -1;
    }
```

```
  else
    {
      if (right == 0)
           right = MAX - 1;
      else
           right = right - 1;
    }
}

void delete_left ()
{
  if (left == -1)
    {
      printf ("\nUNDERFLOW");
      return;
    }
  printf ("\nThe deleted element is %d\n",
  deque[left]); if (left == right)
    {
      left = -1;
      right = -1;
    }
  else
    {
      if (left == MAX - 1)
           left = 0;
      else
           left = left + 1;
    }
}
```

```
void display ()
{
  int front = left, rear = right;
  if (front == -1)
   {
     printf ("\nQueue is Empty\n");
     return;
   }
  printf ("\nThe elements in the queue are:
  "); if (front <= rear)
   {
     while (front <= rear)
          {
            printf ("%d\t", deque[front]);
            front++;
          }
   }

  else
   {
     while (front <= MAX - 1)
          {
            printf ("%d\t", deque[front]);
            front++;
          }

     front = 0;
     while (front <= rear)
          {
            printf ("%d\t", deque[front]);
```

```
                front++;
              }
            }
          printf ("\n");
        }
```

Output :

```
1.Insert at right       2.Insert at left        3.Delete from right
4.Delete from left      5.Display the queue     6.Exit the session
Enter your choice 2

Enter the value to be added 3
1.Insert at right       2.Insert at left        3.Delete from right
4.Delete from left      5.Display the queue     6.Exit the session
Enter your choice 2

Enter the value to be added 4
1.Insert at right       2.Insert at left        3.Delete from right
4.Delete from left      5.Display the queue     6.Exit the session
Enter your choice 1

Enter the value to be added 2
1.Insert at right       2.Insert at left        3.Delete from right
4.Delete from left      5.Display the queue     6.Exit the session
Enter your choice 5

The elements in the queue are: 4        3       2
1.Insert at right       2.Insert at left        3.Delete from right
4.Delete from left      5.Display the queue     6.Exit the session
Enter your choice 3

The deleted element is 2
1.Insert at right       2.Insert at left        3.Delete from right
4.Delete from left      5.Display the queue     6.Exit the session
Enter your choice 4

The deleted element is 4
1.Insert at right       2.Insert at left        3.Delete from right
4.Delete from left      5.Display the queue     6.Exit the session
Enter your choice 5

The elements in the queue are: 3
1.Insert at right       2.Insert at left        3.Delete from right
4.Delete from left      5.Display the queue     6.Exit the session
Enter your choice 6


...Program finished with exit code 0
Press ENTER to exit console.
```

Queue implementation :

Code :

```
#define MAX 5
#include <stdio.h>
#include <stdlib.h>
struct queue
{
   int R, F;
   int data[MAX];
};

void init(struct queue *q)
{
   q->R = -1;
   q->F = -1;
}

int empty(struct queue
*q) {
   if (q->F == -1)
      return (1);
   return (0);
}

int full(struct queue *q)
{
   if (q->R == MAX - 1)
      return (1);
   return (0);
}
```

```
int enqueue(struct queue *q, int
x) {
    if (q->R == -1)
    {
        q->R = q->F = 0;
        q->data[q->R] = x;
    }
    else
    {
        q->R = q->R + 1;
        q->data[q->R] = x;
    }
    return x;
}

int dequeue(struct queue
*q) {
    int x;
    if (q->F == q->R)
    {
        x = q->data[q->F];
        q->F = q->R = -1;
    }
    else
    {
        x = q->data[q->F];
        q->F = q->F + 1;
    }
    return (x);
}
```

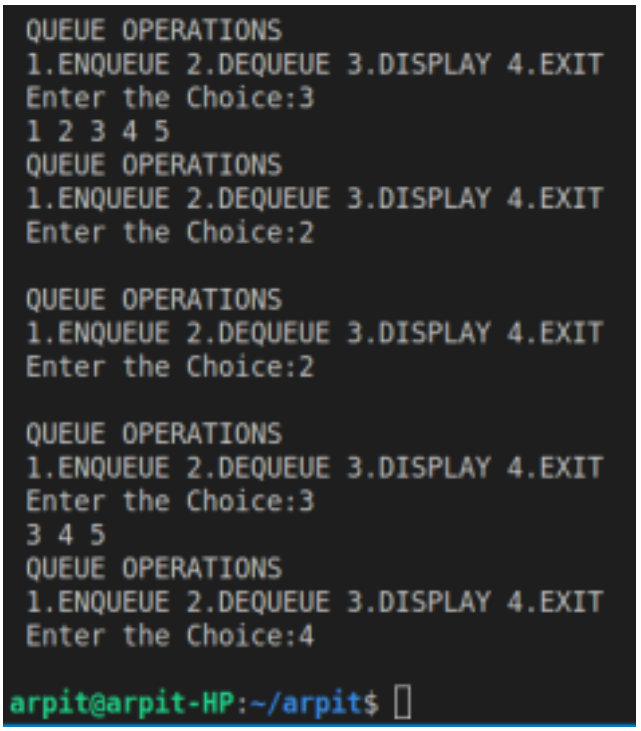
```
void display(struct queue *q)
{
    for (int i = q->F; i <= q->R; i++)
    {
        printf("%d", q->data[i]);
    }
}

int main()
{
    struct queue p;
    init(&p);
    int x;
    int choice;
    do
    {
        printf("\n QUEUE OPERATIONS");
        printf("\n 1.ENQUEUE 2.DEQUEUE 3.DISPLAY
        4.EXIT"); printf("\n Enter the Choice:");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
        {
            if (!full(&p))
            {
                printf("enter value: ");
                scanf("%d", &x);

                enqueue(&p, x);
```

```
    }
    else
    {
        printf("queue is full");
    }
    break;
}
case 2:
{
    if (!empty(&p))
    {
      dequeue(&p);
    }
    else
    {
        printf("queue is empty");
    }
    break;
}
case 3:
{
    display(&p);
    break;
}
case 4:
{
    printf("\n EXIT POINT ");
    break;
}
default:
{
```

```
            printf("\n\t Please Enter a Valid
          Choice(1/2/3/4)"); }
          }
        } while (choice != 4);
        return 0;
      }
```

Output :

```
cd "/home/arpit/arpit/" && gcc queue.c

 QUEUE OPERATIONS
 1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
 Enter the Choice:1
enter value: 1

 QUEUE OPERATIONS
 1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
 Enter the Choice:1
enter value: 2

 QUEUE OPERATIONS
 1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
 Enter the Choice:1
enter value: 3

 QUEUE OPERATIONS
 1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
 Enter the Choice:1
enter value: 4

 QUEUE OPERATIONS
 1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
 Enter the Choice:1
enter value: 5
```



Circular Queue Implementation :

Code :

```
#define MAX 5

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


struct queue

{

   int R, F;

   int data[MAX];

};


void init(struct queue *q)

{

   q->R = -1;

   q->F = -1;

}


int empty(struct queue
```

```
*q) {
    if (q->F == -1)
        return (1);
    return (0);
}

int full(struct queue *q)
{
    if ((q->R+1)%MAX ==
        q->F) return (1);
    return (0);
}

int enqueue(struct queue *q, int
x) {
    if (q->R == -1)
    {
        q->R = q->F = 0;
        q->data[q->R] = x;
    }
    else
    {
        q->R = (q->R +
        1)%MAX; q->data[q->R]
        = x;
    }
    return x;
}

int dequeue(struct queue
*q) {
```

```
    int x;
    if (q->F == q->R)
    {
        x = q->data[q->F];
        q->F = q->R = -1;
    }
    else
    {
        x = q->data[q->F];
        q->F = (q->F +
    1)%MAX; }
    return (x);
}

void display(struct queue *q)
{
    for (int i=q->F;i <= q->R; i++)
    {
        printf("%d", q->data[i]);
        i=(i+1)%MAX;
    }
}

int main()
{
    struct queue p;
    init(&p);
    int x;
    int choice;
    do
    {
```

```
printf("\n QUEUE OPERATIONS");
printf("\n 1.ENQUEUE 2.DEQUEUE 3.DISPLAY
4.EXIT"); printf("\n Enter the Choice:");
scanf("%d", &choice);
switch (choice)
{
case 1:
{
   if (!full(&p))
   {
      printf("enter value: ");
      scanf("%d", &x);
      enqueue(&p, x);
   }
   else
   {
      printf("queue is full");
   }
   break;
}
case 2:
{
   if (!empty(&p))
   {
     dequeue(&p);
   }
   else
   {
      printf("queue is empty");
   }
   break;
```

```
            }
            case 3:
            {
               display(&p);
               break;
            }
            case 4:
            {
               printf("\n EXIT POINT ");
               break;
            }
            default:
            {
               printf("\n\t Please Enter a Valid
            Choice(1/2/3/4)"); }
            }
         } while (choice != 4);
         return 0;
      }
```

Output :

```
cd "/home/arpit/arpit/" && gcc circular
pit/arpit/"circularqueue

 QUEUE OPERATIONS
 1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice1
enter value1

 QUEUE OPERATIONS
 1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice1
enter value2

 QUEUE OPERATIONS
 1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice1
enter value3

 QUEUE OPERATIONS
 1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice1
enter value4

 QUEUE OPERATIONS
 1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice3

Elements in a Queue are :1,2,3,4,
```