NAME - AKRITI CHOUDHARY

ROLL NUMBER - 2005776

SUBJECT - DSA LAB

DATE - 27/10/2021

CLASS - B14

BRANCH - CSE

Question 1)Write a menu driven program to implement queue operations such as Insert, Delete, Display, whether queue is empty etc by using array.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
typedef struct
{
   int data[MAX];
   int front;
   int rear;
} Queue;
int insert(Queue *q, int v)
{
   if (q->rear == MAX - 1)
   {
      printf("Queue Overflow\n");
      return 1;
   }
   else if (q->rear == -1)
   {
      q->front = q->rear = 0;
      q->data[q->rear] = v;
   }
   else
   {
      q->data[++(q->rear)] = v;
   }
   return 0;
}
int del(Queue *q, int *d)
{
   if (q->front == -1)
   {
      printf("Queue is empty\n");
      return 1;
   }
   else if (q->rear == q->front)
   {
      *d = q->data[q->front];
      q->front = q->rear = -1;
      return 0;
   }
   else
   {
      *d = q->data[q->front];
      q->front++;
      return 0;
   }
}
void init(Queue *q)
{
```

```
    q->front = -1;
    q->rear = -1;
}
void traverse(Queue *q)
{
    if (q->front == q->rear == -1)
        printf("queue is empty\n");
    else
    {
        printf("queue elements are:\n");
        for (int i = (q->front); i <= q->rear; i++)
            printf("%d\n", q->data[i]);
        puts(" ");
    }
}
int main()
{
    Queue q1;
    int d, k, c;
    init(&q1);
    do
    {
        printf("1 - to insert an element in the queue\n");
        printf("2 - to delete an element from the queue\n");
        printf("3 - to traverse the stack\n");
        printf("4 - quit\n\n");
        printf("enter your choice\n");
        scanf("%d", &c);
        switch (c)
        {
        case 1:
        {
            int n;
            printf("enter the number to be insert\n");
            scanf("%d", &n);
            k = insert(&q1, n);
            if (k == 0)
                printf("Number is inserted in the queue\n");
            break;
        }
        case 2:
        {
            k = del(&q1, &d);
            if (k == 0)
                printf("Number is deleted from the queue\n");
            break;
        }
        case 3:
        {
            traverse(&q1);
            break;
        }
        case 4:
            puts("Program Terminated");
```

```
        break;
      }
    } while (c != 4);
}
```

```
PS D:\KIIT_NOTES\2nd year sem_3\dsa_lab\26_10_2021> ./ArrayQueue
1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the stack
4 - quit

enter your choice
1
enter the number to be insert
1
Number is inserted in the queue
1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the stack
4 - quit

enter your choice
1
enter the number to be insert
2
Number is inserted in the queue
1 - to insert an element in the queue
```

```
enter the number to be insert
2
Number is inserted in the queue
1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the stack
4 - quit

enter your choice
3
queue elements are:
1
2

1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the stack
4 - quit

enter your choice
2
Number is deleted from the queue
1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the stack
4 - quit
```

```
queue elements are:
2

1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the stack
4 - quit

enter your choice
2
Number is deleted from the queue
1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the stack
4 - quit

enter your choice
2
Queue is empty
1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the stack
4 - quit

enter your choice
4
Program Terminated
```

Question 2)Write a menu driven program to implement queue operations such as Insert, Delete, Display, whether queue is empty etc by using dynamic array.

```
#include <stdio.h>
#include <stdlib.h>
typedef struct
{
   int front;
   int rear;
   int data[];
} Queue;
int insert(Queue *q, int v)
{
   if (q->rear == -1)
   {
      q->front = q->rear = 0;
      q->data[q->rear] = v;
   }
   else
   {
      q->data[++(q->rear)] = v;
   }
   return 0;
}
int del (Queue *q, int *d)
{
   if (q->front == -1)
   {
      printf("Queue is empty");
      return 1;
   }
   else if (q->rear == q->front)
   {
      *d = q->data[q->front];
      q->front = q->rear = -1;
      return 0;
   }
   else
   {
      *d = q->data[q->front];
      q->front++;
      return 0;
   }
}
void init(Queue *q)
{
   q->front = -1;
   q->rear = -1;
}
void traverse(Queue *q)
{
   if (q->front == q->rear == -1)
```

```
      printf("queue is empty\n");
   else
   {
      printf("queue elements are:\n");
      for (int i = (q->front); i <= q->rear; i++)
         printf("%d\n", q->data[i]);
   }
}
int main()
{
   struct Queue *data = malloc(sizeof(int));
   Queue q1;
   int d, k, c;
   init(&q1);
   do
   {
      printf("1 - insert an element in the queue\n");
      printf("2 - delete an element from the queue\n");
      printf("3 - traverse the stack\n");
      printf("4 - quit\n");
      printf("enter your choice\n");
      scanf("%d", &c);
      switch (c)
      {
      case 1:
      {
         int n;
         printf("enter the number you want to insert\n");
         scanf("%d", &n);
         k = insert(&q1, n);
         if (k == 0)
            printf("number is inserted in the queue\n");
         break;
      }
      case 2:
      {
         k = del (&q1, &d);
         if (k == 0)
            printf("%d is deleted from the queue\n", d);
         break;
      }
      case 3:
      {
         traverse(&q1);
         break;
      }
      case 4:
         break;
      }
   } while (c != 4);
}
```

```
PS D:\KIIT_NOTES\2nd year sem_3\dsa_lab\26_10_2021> ./ArrayQueue
1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the stack
4 - quit

enter your choice
1
enter the number to be insert
1
Number is inserted in the queue
1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the stack
4 - quit

enter your choice
1
enter the number to be insert
2
Number is inserted in the queue
1 - to insert an element in the queue
```

```
enter the number to be insert
2
Number is inserted in the queue
1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the stack
4 - quit

enter your choice
3
queue elements are:
1
2

1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the stack
4 - quit

enter your choice
2
Number is deleted from the queue
1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the stack
4 - quit
```

```
queue elements are:
2

1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the stack
4 - quit

enter your choice
2
Number is deleted from the queue
1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the stack
4 - quit

enter your choice
2
Queue is empty
1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the stack
4 - quit

enter your choice
4
Program Terminated
```

Question 3)Write a menu driven program to implement queue operations such as Insert, Delete, Display, whether queue is empty etc by using linked list.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int data;
   struct node *next;
};
typedef struct
{
   struct node *front;
   struct node *rear;
} Queue;
int insert(Queue *q, int v)
{
   struct node *cur = (struct node *)malloc(sizeof(struct node));
   if (cur == NULL)
   {
      return 1;
   }
   cur->data = v;
   cur->next = NULL;
   if (q->front == NULL)
   {
      q->front = q->rear = cur;
   }
   else
   {
      q->rear->next = cur;
      q->rear = cur;
   }
   return 0;
}
int delete (Queue *q, int *d)
{
   if (q->front == NULL)
   {
      printf("Queue underflow\n");
      return 1;
   }
   else
   {
      struct node *temp = q->front;
      if (q->rear == q->front)
      {
         *d = q->front->data;
         q->front = NULL;
         temp == NULL;
         return 0;
```

```
      }
      else
      {
         *d = q->front->data;
         q->front = q->front->next;
         temp == NULL;
         return 0;
      }
   }
}
void init(Queue *q)
{
   q->front = NULL;
   q->rear = NULL;
}
void traverse(Queue *q)
{
   struct node *temp;
   if (q->front == NULL)
      printf("queue is empty\n");
   else
   {
      printf("queue elements are:\n");
      for (temp = (q->front); temp->next != NULL; temp = temp->next)
         printf("%d\n", temp->data);
      printf("%d\n", temp->data);
      puts(" ");
   }
}
int main()
{
   Queue q1;
   int d, k, c;
   init(&q1);
   do
   {
      printf("1 - insert an element in the queue\n");
      printf("2 - delete an element from the queue\n");
      printf("3 - traverse the stack\n");
      printf("4 - quit\n");
      printf("enter your choice\n");
      scanf("%d", &c);
      switch (c)
      {
      case 1:
      {
         int n;
         printf("enter the number to insert\n");
         scanf("%d", &n);
         k = insert(&q1, n);
         if (k == 0)
            printf("number is inserted in the queue\n");
         break;
      }
```

```
    case 2:
    {
       k = delete (&q1, &d);
       if (k == 0)
          printf("%d is deleted from the queue\n", d);
       break;
    }
    case 3:
    {
       traverse(&q1);
       break;
    }
    case 4:
       break;
    }
  } while (c != 4);
}
```

```
PS D:\KIIT_NOTES\2nd year sem_3\dsa_lab\26_10_2
1 - insert an element in the queue
2 - delete an element from the queue
3 - traverse the stack
4 - quit
enter your choice
1
enter the number to insert
9
number is inserted in the queue
1 - insert an element in the queue
2 - delete an element from the queue
3 - traverse the stack
4 - quit
enter your choice
1
enter the number to insert
8
number is inserted in the queue
1 - insert an element in the queue
2 - delete an element from the queue
3 - traverse the stack
4 - quit
enter your choice
3
queue elements are:
```

```
queue elements are:
9
8

1 - insert an element in the queue
2 - delete an element from the queue
3 - traverse the stack
4 - quit
enter your choice
2
9 is deleted from the queue
1 - insert an element in the queue
2 - delete an element from the queue
3 - traverse the stack
4 - quit
enter your choice
2
8 is deleted from the queue
1 - insert an element in the queue
2 - delete an element from the queue
3 - traverse the stack
4 - quit
enter your choice
3
queue is empty
1 - insert an element in the queue
2 - delete an element from the queue
```

```
2 - delete an element from the queue
3 - traverse the stack
4 - quit
enter your choice
4
```

Question 4)Write a menu driven program to implement circular queue operations such as Insert, Delete, Display, whether queue is empty etc by using array.

```
#include <stdio.h>
#define MAX 5
typedef struct
{
   int data[MAX];
   int f;
   int R;
} Cqueue;

void init(Cqueue *q1);
int insert(Cqueue *q, int v);
int del(Cqueue *q, int *d);

int main()
{
   int ch = 0;
   while (ch != 4)
   {
      puts("Enter your choice");
      puts("1 - Insert");
      puts("2 - Delete");
      puts("3 - Display");
      puts("4 - Exit");
      scanf("%d", &ch);

      Cqueue q1;
      init(&q1);
      switch (ch)
      {
      case 1:
      {
         int val;
         puts("Enter the value to be inserted");
         scanf("%d", &val);
         int k = insert(&q1, val);
         if (k == 1)
         {
            puts("Queue overflow");
         }
         else
         {
            puts("Value is inserted in queue");
         }
         break;
      }

      case 2:
      {
         int d = 0;
```

```
            int p = del(&q1, &d);
            if (p == 1)
            {
               puts("Queue underflow");
            }
            else
            {
               puts("Value is deleted from the queue");
            }
            break;
         }
         case 3:
         {
         }
         case 4:
         {
            puts("Program Terminated");
            break;
         }
         default:
         {
            puts("Invalid choice");
            break;
         }
         }
   }
   return 0;
}

void init(Cqueue *q1)
{
   q1->f = -1;
   q1->R = -1;
}

int insert(Cqueue *q, int v)
{
   if (q->f == (q->R + 1) % MAX)
      return 1;
   if (q->R == -1)
   {
      q->f = q->R = 0;
      q->data[q->R] = v;
   }
   else
   {
      q->R = (q->R + 1) % MAX;
      q->data[q->R] = v;
   }
   return 0;
}

int del(Cqueue *q, int *d)
{
```

```
  if (q->f == -1)
     return 1;
  if (q->f == q->R)
  {
     *d = q->data[q->f];
     q->f = q->R = -1;
  }
  else
  {
     *d = q->data[q->f];
     q->f = (q->f + 1) % MAX;
  }
}
```

```
PS D:\KIIT_NOTES\2nd year sem_3\dsa_lab\26_10_202
1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the queue
4 - quit
enter your choice
1
enter the number you want to insert
2
properly inserted
1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the queue
4 - quit
enter your choice
1
enter the number you want to insert
2
properly inserted
1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the queue
4 - quit
enter your choice
3
queue elements are:
2
```

```
queue elements are:
2
2

1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the queue
4 - quit
enter your choice
2
properly deleted 2
1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the queue
4 - quit
enter your choice
2
properly deleted 2
1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the queue
4 - quit
enter your choice
3
queue elements are:
2

1 - to insert an element in the queue
2 - to delete an element from the queue
3 - to traverse the queue
4 - quit
enter your choice
4
Program Terminated
PS D:\KIIT_NOTES\2nd year sem_3\dsa_lab\26_10_2021>
```

Question 5)WAP to implement the double ended queue using array.

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

int deque_arr[MAX];
int left = -1;
int right = -1;

void insert_right()
{
   int added_item;
   if ((left == 0 && right == MAX - 1) || (left == right + 1))
   {
      printf("Queue Overflow\n");
      return;
   }
   if (left == -1)
   {
      left = 0;
      right = 0;
   }
   else if (right == MAX - 1)
      right = 0;
   else
      right = right + 1;
   puts("Enter the element : ");
   scanf("%d", &added_item);
   deque_arr[right] = added_item;
}

void insert_left()
{
   int added_item;
   if ((left == 0 && right == MAX - 1) || (left == right + 1))
   {
      puts("Queue Overflow ");
      return;
   }
   if (left == -1)
   {
      left = 0;
      right = 0;
   }
   else if (left == 0)
      left = MAX - 1;
   else
      left = left - 1;
   puts("Enter the element : ");
   scanf("%d", &added_item);
   deque_arr[left] = added_item;
}
```

```
void delete_left()
{
   if (left == -1)
   {
      puts("Queue Underflow\n");
      return;
   }
   printf("Element deleted from queue is : %d\n", deque_arr[left]);
   if (left == right)
   {
      left = -1;
      right = -1;
   }
   else if (left == MAX - 1)
      left = 0;
   else
      left = left + 1;
}

void delete_right()
{
   if (left == -1)
   {
      puts("Queue Underflow\n");
      return;
   }
   printf("Element deleted from queue is : %d\n", deque_arr[right]);
   if (left == right)
   {
      left = -1;
      right = -1;
   }
   else if (right == 0)
      right = MAX - 1;
   else
      right = right - 1;
}

void display_queue()
{
   int front_pos = left, rear_pos = right;
   if (left == -1)
   {
      puts("Queue is empty");
      return;
   }
   puts("Queue elements :");
   if (front_pos <= rear_pos)
   {
      while (front_pos <= rear_pos)
      {
         printf("%d ", deque_arr[front_pos]);
         front_pos++;
      }
```

```
    }
    else
    {
        while (front_pos <= MAX - 1)
        {
            printf("%d ", deque_arr[front_pos]);
            front_pos++;
        }
        front_pos = 0;
        while (front_pos <= rear_pos)
        {
            printf("%d ", deque_arr[front_pos]);
            front_pos++;
        }
    }
    printf("\n");
}

void input_que()
{
    int choice;
        do
        {       printf("1.Insert at right\n");
                printf("2.Delete from left\n");
                printf("3.Delete from right\n");
                printf("4.Display\n");
                printf("5.Quit\n");
                printf("Enter your choice : ");
                scanf("%d",&choice);

                switch(choice)
                {       case 1:
                        insert_right();
                        break;
                 case 2:
                        delete_left();
                        break;
                 case 3:
                        delete_right();
                        break;
                 case 4:
                        display_queue();
                        break;
                 case 5:
        break;
                        default:
                                printf("Wrong choice\n");
                        }
        }while(choice!=5);

}

void output_que()
{
```

```
    int choice;
    do
    {
        printf("1.Insert at right\n");
        printf("2.Insert at left\n");
        printf("3.Delete from left\n");
        printf("4.Display\n");
        printf("5.Quit\n");
        printf("Enter your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            insert_right();
            break;
        case 2:
            insert_left();
            break;
        case 3:
            delete_left();
            break;
        case 4:
            display_queue();
            break;
        case 5:
            break;
        default:
            printf("Wrong choice\n");
        }
    } while (choice != 5);
}

main()
{
    int choice;
    printf("1.Input restricted dequeue\n");
    printf("2.Output restricted dequeue\n");
    printf("Enter your choice : ");
    scanf("%d", &choice);
    switch (choice)
    {
    case 1:
        input_que();
        break;
    case 2:
        output_que();
        break;
    default:
        printf("Wrong choice\n");
    }
}
```

```
PS D:\KIIT_NOTES\2nd year sem_3\dsa_lab\26_10_2021> g++ dequeue.c -odequeue
PS D:\KIIT_NOTES\2nd year sem_3\dsa_lab\26_10_2021> ./dequeue
1.Input restricted dequeue
2.Output restricted dequeue
Enter your choice : 1
1.Insert at right
2.Delete from left
3.Delete from right
4.Display
5.Quit
Enter your choice : 1
Enter the element :
2
1.Insert at right
2.Delete from left
3.Delete from right
4.Display
5.Quit
Enter your choice : 1
Enter the element :
3
1.Insert at right
2.Delete from left
3.Delete from right
4.Display
5.Quit
Enter your choice : 4
Queue elements :
2 3
1.Insert at right
2.Delete from left
3.Delete from right
4.Display
5.Quit
Enter your choice : 2
Element deleted from queue is : 2
1.Insert at right
2.Delete from left
3.Delete from right
4.Display
5.Quit
Enter your choice : 4
Queue elements :
3
1.Insert at right
2.Delete from left
3.Delete from right
4.Display
5.Quit
Enter your choice : 3
Element deleted from queue is : 3
1.Insert at right
2.Delete from left
3.Delete from right
4.Display
5.Quit
Enter your choice : 5
PS D:\KIIT_NOTES\2nd year sem_3\dsa_lab\26_10_2021>
```