Department of ICT

Faculty of Technology

University of Ruhuna

Data Structures and Algorithms –ICT 2113 **Level2-Semester 1**

Laboratory Assignment 3

Q1.

```
#include<stdio.h>
#define n 5
int main()
{
   int queue[n],ch=1,front=0,rear=0,i,j=1,x=n;
   //clrscr();
   printf("Queue using Array");
   printf("\n1.Enqueue \n2.Dequeue \n3.Display \n4.Exit");
   while(ch)
   {
      printf("\nEnter the Choice:");
      scanf("%d",&ch);
      switch(ch)
      {
      case 1:
         if(rear==x)
            printf("\n Queue is Full");
         else
         {
            printf("\n Enter no %d:",j++);
             scanf("%d",&queue[rear++]);
         }
         break;
      case 2:
         if(front==rear)
         {
            printf("\n Queue is empty");
         }
         else
         {
            printf("\n Deleted Element is %d",queue[front++]);
            x++;
         }
         break;
      case 3:
         printf("\n Queue Elements are:\n ");
         if(front==rear)
            printf("\n Queue is Empty");
```

```
                else
                {
                    for(i=front; i<rear; i++)
                    {
                        printf("%d",queue[i]);
                        printf("\n");
                    }
                    break;
                case 4:
                default:
                    printf("Wrong Choice: please see the options");
                }
            }
        }
```

Q2.

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

#define MAX 5

int queue_array[MAX];
int front = -1;
int rear = -1;
int element;

void enqueue(int element);
int dequeue();
int isEmpty();
int isFull();
int peek();
void display();

int main()
{

    int option;
    while(1)
    {
        printf("\n1. Insert Element in Queue");
        printf("\n2. Delete Element from Queue");
        printf("\n3. Display All the Elements of Queue");
        printf("\n4. Display Element at the Front position");
        printf("\nEnter your option:\t");
        scanf("%d", &option);
        switch(option)
        {
            case 1: printf("\nEnter Element to be Inserted:\t");
                scanf("%d", &element);
```

```
                        enqueue(element);
                        break;

                case 2: element = dequeue();
                        printf("\nDeleted Element From Queue:\t%d", element);
                        break;

                case 3: display();
                        break;

                case 4: printf("\nElement at Front of Queue:\t%d", peek());
                        break;

                case 5: exit(1);
            }
        }
        printf("\n");
}

void enqueue(int element)
{
        if(isFull())
        {
                printf("\nQueue Overflow\n");
                return;
        }
        else if(front == -1)
        {
                front = 0;
        }
        rear = rear + 1;
        queue_array[rear] = element;
```

```
int dequeue()
{
	int item;
	if(isEmpty())
	{
		printf("\nQueue Underflow\n");
		return 1;
	}
	else
	{
		item = queue_array[front];
		front = front + 1;
		return item;
	}
}


int isEmpty()
{
	if(front == -1 || front == rear + 1)
		return 1;
	else
		return 0;
}

int isFull()
{
	if(rear == MAX - 1)
		return 1;
	else
		return 0;
}

int peek()
{
	if(isEmpty())
	{
		printf("\nQueue Underflow\n");
		exit(1);
	}
	else
	{
		return queue_array[front];
	}
}
```

```
void display()
{
	int count;
	printf("\nQueue:\n");
	for(count = front; count <= rear; count++)
	{
		printf("%d\t", queue_array[count]);
	}
}
```

Q3.

```
#include<stdio.h>
#define Buffer_size 4
typedef enum {True=1, False=0} Boolean;

typedef struct {
		char  B_id[20];
		int B_capacity;
		int B_items;
}B_schedule;

typedef struct {
	B_schedule item[Buffer_size];
	 int rear, front;
	int size;
}B_queue;

void init(B_queue *q);
void enqueue (B_queue *q, B_schedule x);
void dequeue (B_queue *q);
Boolean isEmpty(B_queue *q);
Boolean isFull(B_queue *q);
int increment (int x);
void display (B_queue  *q);

int main(){


  B_queue q;
  B_schedule a;
  int i;
  init(&q);
   for (i=0;i<3; i++)
   { printf("Enter Buffer ID :",i+1);
```

```
    scanf("%s", a.B_id);
    printf("Enter Buffer Capacity");
    scanf("%d",&a.B_capacity);
    printf("Enter number of Items");
    scanf("%d",&a.B_items);

  }
  enqueue(&q, a);
  display(&q);

 return 0;
}
 void init(B_queue *q)
 { q->front=0;
   q->rear=-1;
   q->size=0;
 }

  void enqueue (B_queue  *q, B_schedule x)
  { if (isFull(q) )
    printf("Queue is full");
    else
            { q->rear=increment(q->rear);
              q->item[q->rear]= x;
                    q->size++;
                    }
  }

void dequeue (B_queue  *q)
 { if (isEmpty(q) )
     printf("Queue is Empty");
  else
  { q->front= increment(q->front);
    q->size--;

            }
 }
Boolean isEmpty(B_queue  *q)
{ if (q->size==0)
  return True;
  else
    return False;
}
Boolean isFull(B_queue  *q)
{ if (q->size==Buffer_size)
    return True;
```

```
   else
     return False;

}
int increment (int x)
 {
 if (++x== Buffer_size)
     x=0;
   return x;
 }

void display (B_queue  *q)
   { int i;
      printf ("\n Display last Buffer Queue \n ---------- \n");
      for(i=q->front;i<=q->rear;i++){
             printf("Buffer Id  :%s\n",q->item[i].B_id);
            printf("Buffer Capacity  :%d\n",q->item[i].B_capacity);
            printf("Buffer Items :%d\n",q->item[i].B_items); }
               }
```