Department of ICT
Faculty of Technology
University of Ruhuna

Data Structures and Algorithms – ICT2113 **Level 2- Semester 1**

Laboratory Assignment 3 **| 2022**

Objectives

Main objective of this lab session is to get hands on experiences of Queues.

- A queue is a container of objects (a linear collection).
- That are inserted and removed according to the first-in first-out (FIFO) principle.

1. Following C program is written to implement a queue. Some parts of the program is not completed. Complete the program.

```
#include<stdio.h>
#include<stdlib.h>
#define n 5
int main()
{
   int queue[n],ch=1,front=0,rear=0,i,j=1,x=n;
   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:
         //Check Queue is full or not. If not insert an element
         break;
      case 2:
        // Check Queue is Empty or not. If not delete the element
         break;
      case 3:
         printf("\n Queue Elements are:\n ");
          // Check Queue is empty or not. If not, display all elements of the queue
           break;
         case 4:
           exit(0);
```

```
            default:
               printf("Wrong Choice: please see the options");
            }
         }
      }

   }
```

Modify the above program to do the followings

a) Create queue of 8 cells.
b) Insert 5 items into a queue at once.
c) Display the queue
d) Remove 3 items.
e) Display the queue.
f) Insert 5 more items
g) Display the queue/

2. Write a simple c program to implement the queue data structure. Implement the functions

```
#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();

void 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");
    }
```

```
// Function Prototypes
void enqueue(int element)
int dequeue()
int isEmpty()
int isFull()
int peek()
void display()
```

3. Suppose details of I/O scheduling are to be placed in a **bufferQueue** and each **bufferschedule** contains details regarding the **buffer id**, **capacity of buffer** and **number of items in the buffer**.

    - Define a type definition called **bufferschedule** which has fields for **buffer id**, **capacity of buffer** and **number of items in the buffer**
    - Implement a Queue which stores instances of the **bufferQueue** type**.**
    - Implement functions to insert and remove elements from the **bufferschedule**.
    - Implement a function to display the elements of the **bufferschedule**.
    - Define the **bufferschedule** size to 5. Insert three elements to the **bufferschedule**.
    - Display the last buffer queue.
4. Following program demonstrate a Circular queue. Type and run the program. Observe the output.

```
#include<stdio.h>
#include<conio.h>
#define SIZE 5

void enQueue(int);
void deQueue();
void display();

int cQueue[SIZE], front = -1, rear = -1;

void main()
{
  int choice, value;
  clrscr();
  while(1){
    printf("\n****** MENU ******\n");
    printf("1. Insert\n2. Delete\n3. Display\n4. Exit\n");
    printf("Enter your choice: ");
    scanf("%d",&choice);
    switch(choice){
        case 1: printf("\nEnter the value to be insert:  ");
                scanf("%d",&value);
                enQueue(value);
                break;
        case 2: deQueue();
                break;
        case 3: display();
                break;
        case 4: exit(0);
        default: printf("\nPlease select the correct choice!!!\n");
    }
  }
}
void enQueue(int value)
{
  if((front == 0 && rear == SIZE - 1) || (front == rear+1))
    printf("\nCircular Queue is Full! Insertion not possible!!!\n");
  else{
    if(rear == SIZE-1 && front != 0)
        rear = -1;
    cQueue[++rear] = value;
    printf("\nInsertion Success!!!\n");
    if(front == -1)
        front = 0;
  }
```

```
}
void deQueue()
{
  if(front == -1 && rear == -1)
    printf("\nCircular Queue is Empty! Deletion is not possible!!!\n");
  else{
    printf("\nDeleted element : %d\n",cQueue[front++]);
    if(front == SIZE)
        front = 0;
    if(front-1 == rear)
        front = rear = -1;
  }
}
void display()
{
  if(front == -1)
    printf("\nCircular Queue is Empty!!!\n");
  else{
    int i = front;
    printf("\nCircular Queue Elements are : \n");
    if(front <= rear){
        while(i <= rear)
          printf("%d\t",cQueue[i++]);
    }
    else{
        while(i <= SIZE - 1)
          printf("%d\t", cQueue[i++]);
        i = 0;
        while(i <= rear)
          printf("%d\t",cQueue[i++]);
    }
  }
}
```