**Kathmandu University**

**Department of Computer Science and Engineering**

**Dhulikhel, Kavre**



**A Report on**

**COMP 202: Data Structures and Algorithms**

**Mini Project**

**Submitted by:**

Bisheshwor Neupane(35)

Gaurab Shrestha(43)

Umesh Shrestha(47)

**Submitted to:**

Dr. Rajani Chulyadyo

Department of Computer Science and Engineering

**Submission Date:** January 15, 2019

**Task:**  Implement a queue using a circularly linked list with the following operations: • enqueue – Inserts an element at the end of the queue

• dequeue – Displays and deletes an element from the front of the queue

• rear – Displays the last element of the queue • front – Displays the first element of the queue

• isEmpty – Returns true if the queue is empty and false otherwise

• isFull – Returns true if the queue is full and false otherwise

What are the time complexities of these operations in your implementation? You are supposed to use the linked list implementation done in Lab 2.

Implementation:

Algorithm

Time Complexity of the program

|  |  |  |
| --- | --- | --- |
| S.N | Program | Steps |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11.  12.  13.  14.  15.  16.  17.  18.  19.  20.  21.  22.  23.  24.  25.  26. | int main(){  cout<<"\nQueue Implementation:"<<endl;  Queue qu;  qu.isEmpty();  qu.enqueuetohead(4);  qu.dequeue();  for(int i=1;i<15;i=i+2){  if(qu.isFull() == false)  qu.enqueue(i);  else{  cout<<"Queue is full !";  break;  }  }  //qu.isFull();  if(qu.isFull() == true)  cout<<" queue full ";  else  cout<<"Not full ";  //qu.dequeue();  qu.isEmpty();  qu.showitems();  cout<<"\n Rear and front:";  cout<<qu.rear()<<" ";  cout<< qu.front();  return 0;  } |  |
|  |  |  |