**Lab Manual**

**Department of Computer Science Session 2019 (Spring-2020)**

**Data Structure**

This course is a comprehensive introductory course that is intended for students who have prerequisite knowledge of basic computer programming as C, C++ etc. The aim of this course is to provide students with a firm foundation of “Data Structure” and to make them able to design C++ code of algorithms and different problems. The aim is to provide foundation of data structure and to polish the skills of understanding computational problems in algorithmic way. The programming language used in this course is C++.

**Lab Manual**

**Department of Computer Science Session 2019 (Spring-2020)**

**Data Structure**

***(Lab XXX )***

**Target:** Implementation of Queue using Array and Linked List.

1. **Guidelines/Instructions:**

* Be attentive, honest and prepare logic for given problem on your own
* Programming assignments in LAB require you to write and run a computer program.
* Comments should be used to give the statement of the problem and every function should indicate the purpose of the function, inputs, and outputs
* Statements within the program should be properly indented and use meaningful names for variables and functions.
* Plagiarism/Cheating is highly discouraged by assigning 0 to both who tried and one who shared his/her code.
* Some LAB tasks, assignments count toward your final course grade, while others are just for practice.
* Lab work must be evaluated during lab timing and homework is for submission.
* Contact your teacher and GAs during their office hours (mentioned at end of lab.)

**Reading Material:**

* Consult Chapter No 9 “Arrays and Strings” of book “C++: Programming from Problem Analysis to Program Design by D.S Malik (latest edition)” for better understanding of given problems**.**
* C++ Basics :

(<https://www.tutorialspoint.com/cplusplus/cpp_arrays.htm>)

* C++ programming examples and libraries:

(<https://www.programiz.com/cpp-programming/examples>)

* C++ Arrays:

**(**<http://www.cplusplus.com/doc/tutorial/arrays/>)

***Theory: A queue is an abstract data structure that contains a collection of elements. Queue implements the FIFO mechanism i.e. the element that is inserted first is also deleted first. In other words, the least recently added element is removed first in a queue.***

**A program that implements the queue using an array is given as follows −**

**// Example program 1**

#include <iostream>

using namespace std;

int queue[100], n = 100, front = - 1, rear = - 1;

void Insert() {

   int val;

   if (rear == n - 1)

   cout<<"Queue Overflow"<<endl;

   else {

      if (front == - 1)

      front = 0;

      cout<<"Insert the element in queue : "<<endl;

      cin>>val;

      rear++;

      queue[rear] = val;

   }

}

void Delete() {

   if (front == - 1 || front > rear) {

      cout<<"Queue Underflow ";

      return ;

   } else {

      cout<<"Element deleted from queue is : "<< queue[front] <<endl;

      front++;;

   }

}

void Display() {

   if (front == - 1)

   cout<<"Queue is empty"<<endl;

   else {

      cout<<"Queue elements are : ";

      for (int i = front; i <= rear; i++)

      cout<<queue[i]<<" ";

         cout<<endl;

   }

}

int main() {

   int ch;

   cout<<"1) Insert element to queue"<<endl;

   cout<<"2) Delete element from queue"<<endl;

   cout<<"3) Display all the elements of queue"<<endl;

   cout<<"4) Exit"<<endl;

   do {

      cout<<"Enter your choice : "<<endl;

      cin<<ch;

      switch (ch) {

         case 1: Insert();

         break;

         case 2: Delete();

         break;

         case 3: Display();

         break;

         case 4: cout<<"Exit"<<endl;

         break;

         default: cout<<"Invalid choice"<<endl;

      }

   } while(ch!=4);

   return 0;

}

1. **Tasks to be completed in lab:**
2. Write a C++ program that implement the queue using linked list.
3. Write a C++ program to implement queues by the simple method of keeping the head of the queue always in the first position of linear array.
4. Write a C++ function to implement queues in circular array with one unused entry in the array. That is, we consider that the array is full when the rear is two position before the front; when the rear is one position before, it will always indicate an empty queue.
5. If we implement a queue as a circularly linked list then we need only one pointer rear (or tail) to locate both the front and rear. Write a C++ program to process a queue stored in this way.
6. Initialize
7. Add Node
8. Delete Node
9. Write a function that will read one line of input from the terminal. The input is supposed to consist of two parts separated by colon ‘;’. As its result your function should produce a single character as follows:

**N** No colon on the line.

**L** The left part (before the colon) is longer than the right

**R** The right part (after the colon) is longer than the left**.**

**D** The left and right parts have same length but are different.

**S** The left and right parts are the same.

Use a queue to keep track of the left part of line while reading the right part.

1. **Convert the following Algorithm into C++ code to implement Queue as Linked List.**

**CREATE**

1. t = new node
2. Enter info to be inserted
3. Read n
4. t Æ info = n
5. t Æ next = front
6. front = t

**INSERTION**

1. r Æ next = t
2. t Æ next = NULL
3. Return

**DELETION**

1. x = front
2. front = front Æ next
3. delnode(x)
4. Return

**DISPLAY**

1. If (front = NULL) Print “ empty queue” Return

Else

P = start

Repeat until (p< > NULL) Print p Æ info

P = pÆ next Return

1. **Tasks to be submitted:**
2. Write a C++ program that implement the queue using Array.

**Note:** Ask the student to list the drawbacks of implementing the queue using array. The following algorithm cold be repeated using structures and linked lists.

***Algorithm:--***

**Data Structure:**

Q is the array representation of queue structure; two pointers FRONT and Rear of the queue Q are known.

* Start
* Initialize the array of 10 elements and name it as queue.
* Initialize other variables like rear and front in the beginning of the program.
* Give the choice to the users for different operations on Queue like insert, Delete, Display AND Exit.
* If the choice = Insert operation call the function **qinser()**
* If the choice = delete operation call the function **qdelete()**
* If the choice = Display operation call the function **qdisplay()**
* If the choice = Exit, then exit from the program > end.

**Function insert()**

* Check for the overflow condition of Queue
  + If(REAR=N) the Print “Queue is full”
* If no overflow, increment the value of rear
  + REAR = REAR+1
* Get the element to be inserted into the queue from the user

Q[REAR] = ITEM > Assign it as the last value, queue[rear]

**Function delete()**

* Check for the underflow(empty) condition of the queue > if not empty, Output the element to be deleted from the queue > increment the value of front.

**Function Display()**

* Display the element of the queue

**Output:**

Enter the operation to be performed:

1) Enqueue

2) Dequeue

3)Display

4) Exit

1

Enter the number to be added 11

Enter the operation to be performed: 1) Enqueue 2) Dequeue 3)Display 4) Exit

1

Enter the number to be added 22

Enter the operation to be performed: 1) Enqueue 2) Dequeue 3)Display 4) Exit

1

Enter the number to be added 33

Enter the operation to be performed: 1) Enqueue 2) Dequeue 3)Display 4) Exit

2

The number to be deleted is 11:

Enter the operation to be performed: 1) Enqueue 2) Dequeue 3)Display 4) Exit

3

The queue is 22 33 44

**Conclusion:**

All the queue operations are successfully performed.

1. **Submission instructions:**

* See manual 1 for understanding “How to submit your assignment”.
* All rules discussed in lab1 for grading assignments will apply for all labs of this course.

1. **Office hours and email address for communication:**

* See lab manual 1