# Department of Computing

# School of Electrical Engineering and Computer Science

**CS-250: Data Structure and Algorithms**

**Class: BSCS 10AB**

**Lab 05 : Queue**

**CLO1: Understand the fundamentals of data structures and algorithms**

**Date: 26th October 2021**

**Time: 10:00 am – 12:50 pm   
&  
 02:00 pm – 4:50 pm**

# Instructor: Dr. Yasir Faheem

# Lab Engineer: Aftab Farooq

# 

# Lab 05 : Linear and Circular Queue

**Introduction**

This lab is based on queues and its implementation statically and dynamically.

**Objectives**

Objective of this lab is to get familiar with the queues and implement it in a programming language

**Tools/Software Requirement**

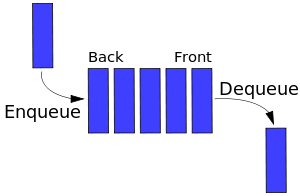
Visual Studio c++, Eclipse C++ IDE

**Helping Material**

Lecture slides, text book

**Description**

In [computer science](http://en.wikipedia.org/wiki/Computer_science), a queue is a particular kind of [abstract data type](http://en.wikipedia.org/wiki/Abstract_data_type) or [collection](http://en.wikipedia.org/wiki/Collection_(computing)) in which the entities in the collection are kept in order and the principal (or only) operations on the collection are the addition of entities to the rear terminal position and removal of entities from the front terminal position. This makes the queue a [First-In-First-Out (FIFO) data structure](http://en.wikipedia.org/wiki/FIFO_(computing)).



The following sets of operation are generally supported by queue.

1. void Enqueue(element) – add an element at the rear end of the queue

2. element Dequeue() – removes and display the element from the front end of the queue

3. bool isEmpty() – checks if the queue is empty or not

4. bool isFull() – checks if the queue is full or not

5. void Clear() – release the memory allocated by queue

6. void FirstElement() – display the contents of first element of queue at front location

**Tasks:**

**Task 1:**

You are required to implement all operations of Queue ADT using an array-based linear queue.

Code:

#include<iostream>

using namespace std;

#define size 10

int \*arry=new int [size];

int head = -1;

int tail = -1;

int length = 0;

bool isEmpty()// checking if empty

{

return head < 0;

}

bool isFull()// checking if queue is full

{

return tail == size;

}

void Enqueue(int value)// add at the tail of the queue

{

if (!isFull())

{

arry[++tail] = value;

length++;

}

else

cout << " List is Full." << endl;

}

int Dequeue()// remove from the head of queue

{

if (!isEmpty())

{

int val;

val = arry[head];

arry[++head] = NULL;

length--;

return val;

}

else

cout << " List is Empty." << endl;

}

void FirstElement()// check what is the first element

{

if (!isEmpty())

{

cout << arry[head] << endl;

}

else

cout << " List is Empty." << endl;

}

void clear()// delete all list

{

delete arry;

cout << " Queue Deleted." << endl;

}

void print()//print all element of queue

{

if (!isEmpty())

{

for (int i = 0; i < size - 1; i++)

{

cout << arry[i] << endl;

}

}

}

int main()

{

Enqueue(4);

Enqueue(7);

Enqueue(9);

cout << Dequeue() << endl;

Enqueue(10);

cout << Dequeue() << endl;

FirstElement();

cout << length << endl;

return 0;

}

Screenshot:

Text

Description automatically generated

**Task 2:**

You are required to implement all operations of Queue ADT using an array-based circular queue

Code:

#include<iostream>

#include<string>

using namespace std;

#define size 500

struct q

{

int data;

string name;

};

q\* arry = new q[size];

int head = 0;

int tail = -1;

int index = -1;

int length = -1;

bool isEmpty()// checking if empty

{

return length < 0;

}

bool isFull() // checking if queue is full

{

return length == size;

}

void Enqueue(int value) // add at the tail of the queue

{

if (!isFull())

{

if (tail != size - 1)

{

arry[++tail].data = value; length++;

}

else

{

tail = 0;

arry[tail].data = value; length++;

}

}

else

cout << " List is Full." << endl;

}

int Dequeue()// remove from the head of queue

{

if (!isEmpty())

{

if (head != size - 1)

{

int val;

val = arry[head].data;

arry[head++].data = -999;

length--;

return val;

}

else

{

int val;

val = arry[head].data;

length--;

arry[head].data = -999;

head = 0;

return val;

}

}

{ cout << " List is Empty." << endl;

return -9999; }

}

void FirstElement()// check what is the first element

{

if (!isEmpty())

{

cout << arry[head].data << endl;

}

else

cout << " List is Empty." << endl;

}

void clear() // delete all list

{

delete arry;

cout << " Queue Deleted." << endl;

}

void print() //print all element of queue

{

if (!isEmpty())

{

for (int i = 0; i < size - 1; i++)

{

cout << arry[i].data << endl;

}

}

}

int main()

{

Enqueue(4);

Enqueue(7);

Enqueue(9);

cout << Dequeue() << endl;

Enqueue(10);

cout << Dequeue() << endl;

FirstElement();

cout << length << endl;

return 0;

}

Screenshot:

Text

Description automatically generated

**Task 3:**

You have to implement a waiting room management system in an emergency ward of a hospital.  Your program will assign an Id number to a patient in a first come first serve basis. The lower the id, the sooner the service will be provided to the patient.

Your program will contain the following methods:

**RegisterPatient():** This method assigns an Id (which is auto-generated) to a patient and register him/her to the system.

**ServePatient():** This method calls a patient to provide hospital service to him/her.

**CancelAll():** This method cancels all appointments of the patients so that the doctor can go to lunch.

**CanDoctorGoHome():** This method returns true if no one is waiting, otherwise, returns false.

**ShowAllPatient():** This method shows all ids of the waiting patients in SORTED order. (Hint: use the sorting methods learnt in class using the appropriate data-structure for each task) [Sorted according to their names]

Solve the above problem using an array based circular queue.

Code:

#include<iostream>

#include<string>

using namespace std;

#define size 500

struct q

{

int data;

string name;

};

q\* arry = new q[size];

int head = 0;

int tail = -1;

int index = -1;

int length = -1;

bool isEmpty()

{

return length < 0;

}

bool isFull()

{

return length == size;

}

void Enqueue(int value)

{

if (!isFull())

{

if (tail != size - 1)

{

arry[++tail].data = value; length++;

}

else

{

tail = 0;

arry[tail].data = value; length++;

}

}

else

cout << " List is Full." << endl;

}

int Dequeue()

{

if (!isEmpty())

{

if (head != size - 1)

{

int val;

val = arry[head].data;

arry[head++].data = -999;

length--;

return val;

}

else

{

int val;

val = arry[head].data;

length--;

arry[head].data = -999;

head = 0;

return val;

}

}

{ cout << " List is Empty." << endl;

return -9999; }

}

void FirstElement()

{

if (!isEmpty())

{

cout << arry[head].data << endl;

}

else

cout << " List is Empty." << endl;

}

void clear()

{

delete arry;

cout << " Queue Deleted." << endl;

}

void print()

{

if (!isEmpty())

{

for (int i = 0; i < size - 1; i++)

{

cout << arry[i].data << endl;

}

}

}

void RegisterPatient()

{

if (!isFull())

{

Enqueue(++index);

cout << " Enter Your Name: "; cin >> arry[index].name;

cout << "Your Number is " << index << endl;

if (index == 0)

cout << " Please Come. Doctor is Free." << endl;

else

cout << " Please Wait Your Turn." << endl;

}

}

void Servepatient()

{

if (!isEmpty())

{

cout << " Currently Serving...Patient Number: " << Dequeue() << endl;

}

}

void CancelAll()

{

cout << " All Appointments Are Cancelled. Doctor Unavailable." << endl;

delete arry;

}

void CanDoctorGoHome()

{

if (isEmpty())

{

cout << " All Patients Served." << endl;

}

else

{

cout << " Number of Patients Left: " << length + 1 << endl;

}

}

void ShowAllPatients()

{

for (int i = head; i <= length; i++)

{

cout << "Index: " << arry[i].data << " Patient Name is " << arry[i].name << endl;

}

}

void Menu()

{

cout << " Options" << endl << " 1.Register Patient" << endl << " 2.Serve Patient " << endl << " 3.Cancel All" << endl <<

" 4.Check For Patients" << endl << " 5.Show All Patients" << endl;

}

int main()

{

int option = -1;

Menu(); cout << " Enter Option: "; cin >> option;

while (option != 0)

{

//case statement to for the processing the options

switch (option)

{

case 1:RegisterPatient(); break;

case 2:Servepatient(); break;

case 3:CancelAll(); break;

case 4:CanDoctorGoHome(); break;

case 5:ShowAllPatients(); break;

default: cout << " Wrong Option Entered." << endl;

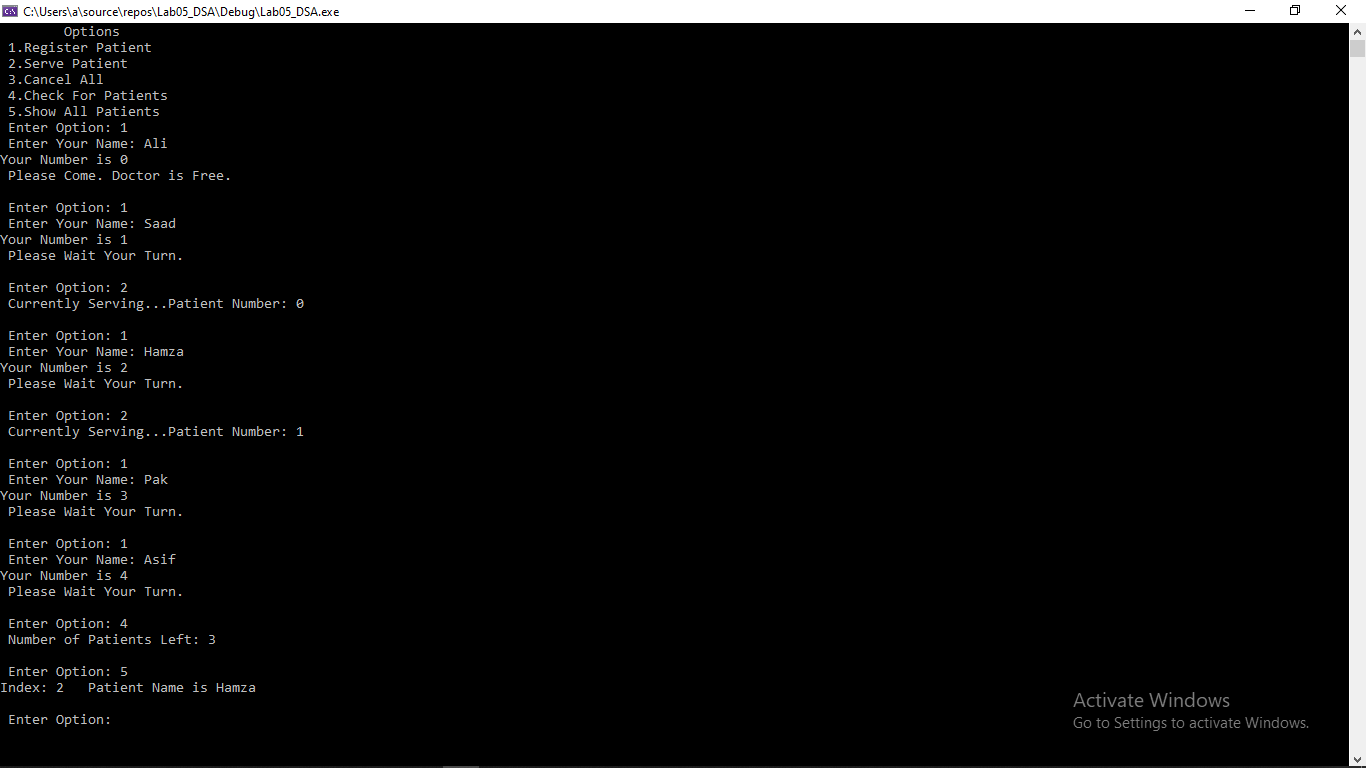
break;

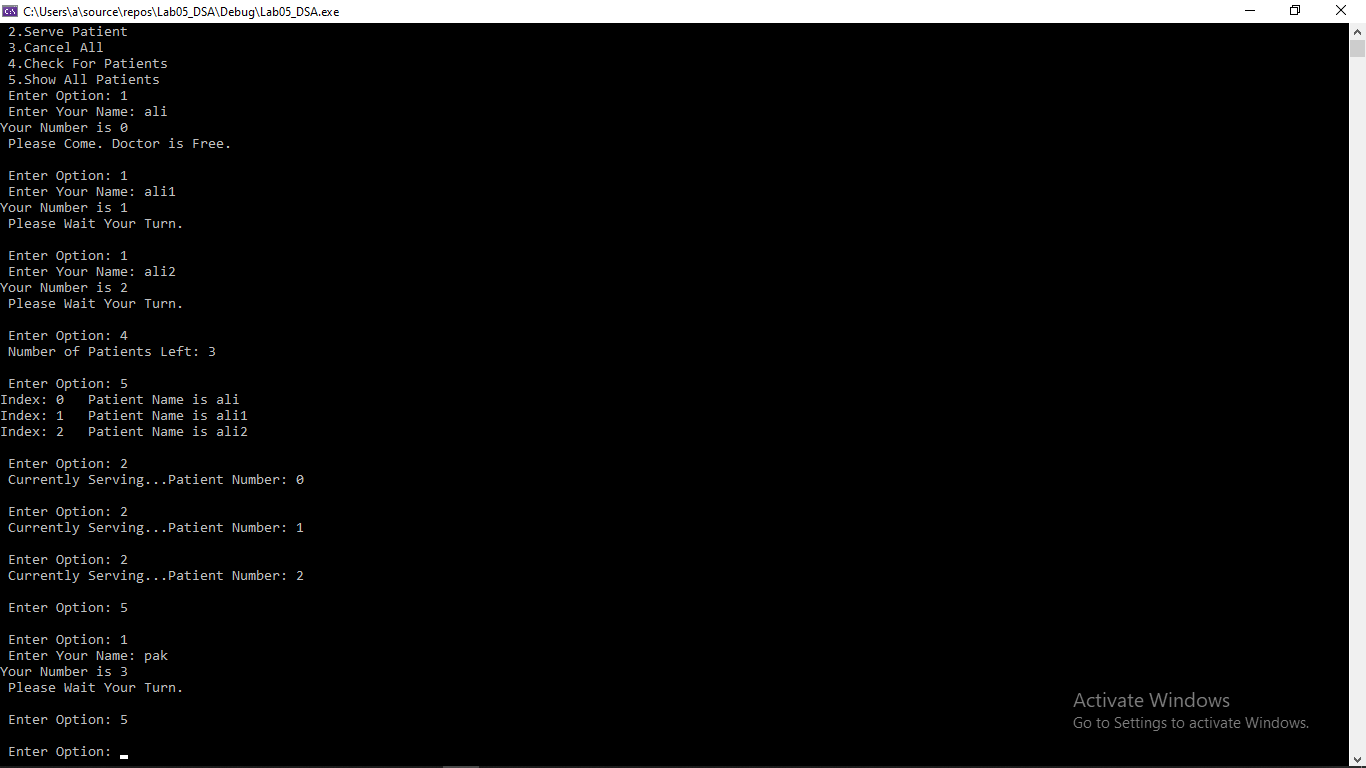
}cout << endl << " Enter Option: "; cin >> option;

}

}

Screenshot:





**Task 4:**

Take an array of non-negative integers. Find the largest multiple of 3 that can be formed from array elements.   
For example, if the input array is {8, 1, 9}, the output should be “9 8 1”, and if the input array is {8, 1, 7, 6, 0}, output should be “8 7 6 0”.

**Hint :**   
1. Sort the array in non-decreasing order.  
2. Take three queues. One for storing elements which on dividing by 3 gives remainder as 0.The second queue stores digits which on dividing by 3 gives remainder as 1. The third queue stores digits which on dividing by 3 gives remainder as 2. Call them as queue0, queue1 and queue2  
3. Find the sum of all the digits.  
4. Three cases arise:   
……4.1 The sum of digits is divisible by 3. Dequeue all the digits from the three queues. Sort them in non-increasing order. Output the array.  
……4.2 The sum of digits produces remainder 1 when divided by 3.   
Remove one item from queue1. If queue1 is empty, remove two items from queue2. If queue2 contains less than two items, the number is not possible.  
……4.3 The sum of digits produces remainder 2 when divided by 3.   
Remove one item from queue2. If queue2 is empty, remove two items from queue1. If queue1 contains less than two items, the number is not possible.  
5. Finally empty all the queues into an auxiliary array. Sort the auxiliary array in non-increasing order. Output the auxiliary array.

Code:

#include<iostream>

#include<string>

using namespace std;

#define size 500

struct q

{

int data;

};

// for each queue make head,tail and length variable

int head1 = 0;

int tail1 = -1;

int length1 = -1;

int head2 = 0;

int tail2 = -1;

int length2 = -1;

int head3 = 0;

int tail3 = -1;

int length3 = -1;

// using three differnt queues in the process

int\* queue1 = new int[size];

int\* queue2 = new int[size];

int\* queue3 = new int[size];

int\* arry4 = new int[size];

bool isEmpty(int arry[],int &length)

{

return length < 0;

}

bool isFull(int arry[],int &length)

{

/\*if (tail != size - 1)

return tail == head + 1;

else

return head == 0;\*/

return length == size;

}

void Enqueue(int value, int arry[], int &tail, int &length)

{

if (!isFull(arry, length))

{

if (tail != size - 1)

{

arry[++tail] = value; length++;

}

else

{

tail = 0;

arry[tail] = value; length++;

}

}

else

cout << " List is Full." << endl;

}

int Dequeue(int arry[],int &head,int &length)

{

if (!isEmpty(arry, length))

{

if (head != size - 1)

{

int val;

val = arry[head];

arry[head++] = -999;

length--;

return val;

}

else

{

int val;

val = arry[head];

length--;

arry[head] = -999;

head = 0;

return val;

}

}

{ cout << " List is Empty." << endl;

return -9999; }

}

void FirstElement(int arry[],int &head,int &length)

{

if (!isEmpty(arry, length))

{

cout << arry[head] << endl;

}

else

cout << " List is Empty." << endl;

}

void clear(int arry[])

{

delete arry;

cout << " Queue Deleted." << endl;

}

void print(int arry[],int &length)

{

if (!isEmpty(arry, length))

{

for (int i = 0; i < size - 1; i++)

{

cout << arry[i] << endl;

}

}

}

void sort\_acc(int arr[],int siz)// sorting array in accending order

{

int temp = 0;

for (int i = 0; i < siz; ++i)

{

for (int j = i + 1; j < siz; ++j)

{

if (arr[i] > arr[j])

{

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

}

void sort\_dcc(int arr[], int siz)// sorting array in decending order

{

int temp = 0;

for (int i = 0; i < siz; ++i)

{

for (int j = i + 1; j < siz; ++j)

{

if (arr[i] < arr[j])

{

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

}

int main()

{

int siz = 0;

int sum = 0;

int len = 0;

length1 = 0;

length2 = 0;

length3 = 0;

cout << " Enter Size of the list: "; cin >> siz;

int\* list = new int[siz];

for (int x = 0; x < siz; x++)// take input from user

{

cout << " Enter Value " << x + 1 << ": "; cin >> list[x]; sum = sum + list[x];

}

sort\_acc(list, siz);// sort in accending order

for (int i = 0; i < siz; i++)

{

cout << list[i] << endl;

}

// applying the checks

for (int x = 0; x < siz; x++)

{

if (list[x] % 3 == 0)

{

Enqueue(list[x], queue1, tail1, length1);

}

else if (list[x] % 3 == 1)

{

Enqueue(list[x], queue2, tail2, length2);

}

else if (list[x] % 3 == 2)

{

Enqueue(list[x], queue3, tail3, length3);

}

}

if (sum % 3 == 0)

{

for (int i = 0; i < length1; i++)

{

arry4[len] = Dequeue(queue1, head1, length1); len++;

}

for (int i = 0; i < length2; i++)

{

arry4[len] = Dequeue(queue2, head2, length2); len++;

}

for (int i = 0; i < length3; i++)

{

arry4[len] = Dequeue(queue3, head3, length3); len++;

}

}

else if (sum % 3 == 1)

{

if (!isEmpty(queue1, length1))

{

arry4[len] = Dequeue(queue1, head1, length1); len++;

}

else

{

if (length2 >= 2)

{

for (int i = 0; i < 2; i++)

{

arry4[len] = Dequeue(queue2, head2, length2); len++;

}

}

else

{

cout << "Number is Not Possible,with this combination. " << endl;

}

}

}

else if (sum % 3 == 2)

{

if (!isEmpty(queue2, length2))

{

arry4[len] = Dequeue(queue2, head2, length2); len++;

}

else

{

if (length1 >= 2)

{

for (int i = 0; i < 2; i++)

{

arry4[len] = Dequeue(queue1, head1, length1); len++;

}

}

else

{

cout << "Number is Not Possible,with this combination. " << endl;

}

}

}

sort\_dcc(arry4, len);

// outputing the largest number

cout << " Largest Multiple of 3: ";

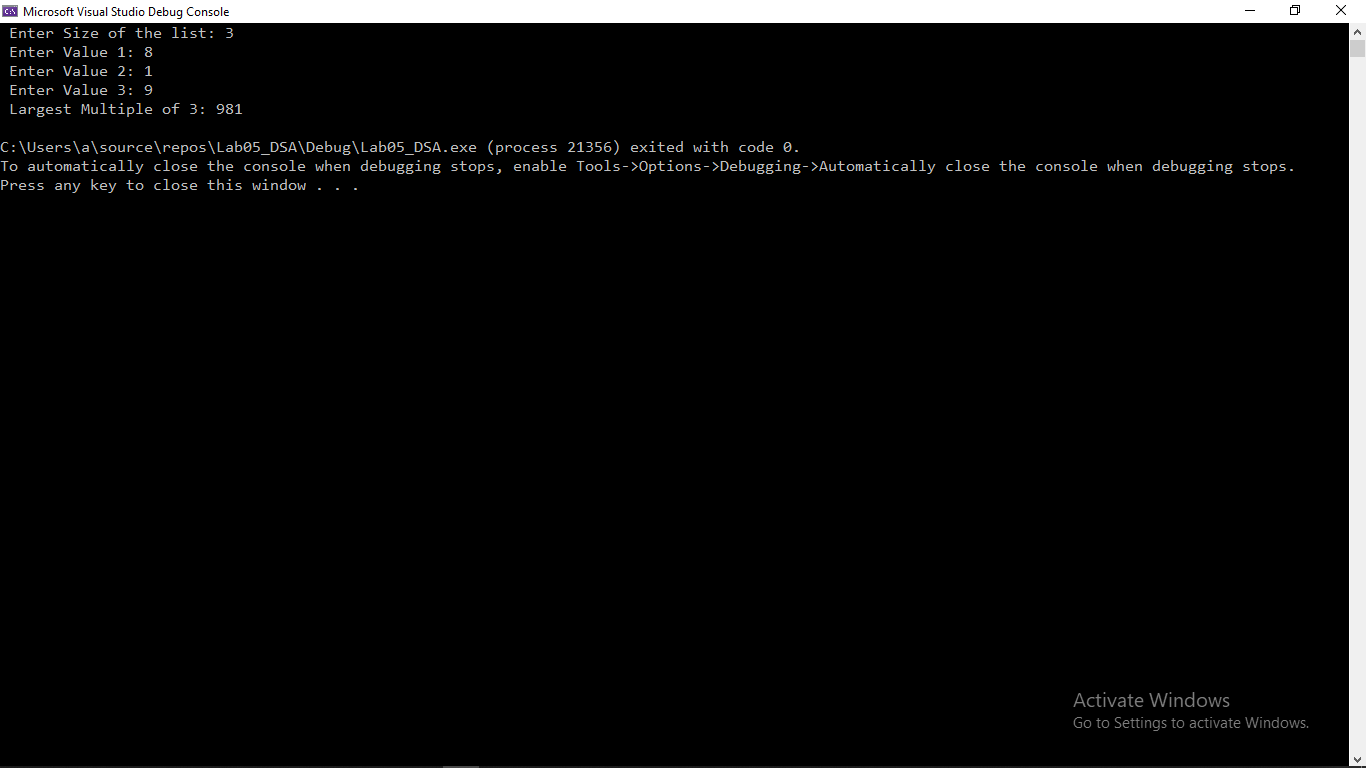
for (int j = 0; j < len; j++)

cout << arry4[j];

cout << endl;

}

Screenshot:



**Deliverables:**

Compile a single word document by filling in the solution part and submit this Word file on LMS. The name of word document should follow this format. i.e. **YourFullName(reg)\_Lab#.** This lab grading policy is as follows: The lab is graded between 0 to 10 marks. The submitted solution can get a maximum of 5 marks. At the end of each lab or in the next lab, there will be a viva related to the tasks. The viva has a weightage of 5 marks. Insert the solution/answer in this document. You must show the implementation of the tasks in the designing tool, along with your complete Word document to get your work graded. You must also submit this Word document on the LMS. In case of any problems discuss it by emailing it to [aftab.farooq@seecs.edu.pk](mailto:aftab.farooq@seecs.edu.pk).

**Note:** Students are required to upload the lab on LMS before deadline.

Use proper indentation and comments. Lack of comments and indentation will result in deduction of marks.