**Module 5 - Queues, Iterators, Applications of Stacks and Queues**

**Overview**

In the last unit you have learned one of the linear data structure called Stacks and the operations (insert, delete and peek) performed on the Stacks. In this unit you are going to learn about Queues and its applications.

A Queue is a linear data structure that follows the first in first out principle. The first element to be inserted into a Queue is the first element to be removed from the Queue.

**Learning Objectives**

1. Working of queues
2. Applications of Stacks and Queues

**Module 5 - Queues, Iterators, Applications of Stacks and Queues**

**Activity4**

**Learning by doing-1**

Suppose that a client performs an intermixed sequence of (queue) enqueue and dequeue operations. The enqueue operations put the integers 0 through 9 in order onto the queue; the dequeue operations print out the return value. Which of the following sequence(s) could not occur?  
(a) 0 1 2 3 4 5 6 7 8 9  
(b) 4 6 8 7 5 3 2 9 0 1   
(c) 2 5 6 7 4 8 9 3 1 0  
(d) 4 3 2 1 0 5 6 7 8 9

* Answer the above problem in a file name problem-1.txt and add it to the folder ADS-1-practice/m5
* Push the respository to GitHub when your answer is done
* Enter the git commit ID in the blank below

Do not use eval to submit this activity

**Learning by doing-2**

What does the following code fragment do to the queue q?

Stack stack = new Stack();

while (!q.isEmpty())

stack.push(q.dequeue());

while (!stack.isEmpty())

q.enqueue(stack.pop());

* Answer the above problem in a file name problem-2.txt and add it to the folder ADS-1-practice/m5
* Push the respository to GitHub when your answer is done
* Enter the git commit ID in the blank below

Do not use eval to submit this activity

**Module 5 - Queues, Iterators, Applications of Stacks and Queues**

**Activity6**

**Assignment-1**

Implement Deque. A double-ended queue or deque (pronounced “deck”) is like a stack or a queue but supports adding and removing items at both ends. Check your program against the given test cases below as a zip file. Submit your Solution(zip file) when all the test cases are passed.

* Download the starter code; the directory structure is similar to the sample-assignment seen in the previous activity
* Add a subfolder m5 for Module 5
* Unzip the starter code into m5 folder. You should see a folder with the name 5.1 Deque
* You should write your solution in the file Solution.java
* There are a few lines of code to handle the input testcases
* After you write the code use eval to check if you got all the testcases right
* submit commit ID in the textbox below.

**Module 5 - Queues, Iterators, Applications of Stacks and Queues**

**Activity7**

**Assignment-2**

In the Josephus problem from antiquity, N people are in dire straits and agree to the following strategy to reduce the population. They arrange themselves in a circle (at positions numbered from 0 to N-1) and proceed around the circle, eliminating every Mth person until only one person is left. Legend has it that Josephus figured out where to sit to avoid being eliminated. Write a Queue client that takes M and N from the command line and prints out the order in which people are eliminated (and thus would show Josephus where to sit in the circle). Check your program against the given test cases below as a zip file. Submit your Solution(zip file) when all the test cases are passed.

* Download the starter code; the directory structure is similar to the sample-assignment seen in the previous activity
* Unzip the starter code into m5 folder. You should see a folder with the name 5.2 Josephus Problem
* You should write your solution in the file Solution.java
* There are a few lines of code to handle the input testcases
* After you write the code use eval to check if you got all the testcases right
* submit commit ID in the textbox below.