

DP Weekly Question #1

Part a.)

In this part, we will be looking at data structures. This problem will use only two data structures, a stack and a queue. A stack is a Last-In-First-Out (LIFO) data structure, meaning that the first element that you can access out of the data structure is the last one you inserted. A stack supports two operations, `push()` and `pop()`. A `push()` pushes an element on to the top of the stack, and a `pop()` takes the top element off the stack and returns it. A queue (pronounced like the letter Q) is a First-In-First-Out (FIFO) data structure. The first element you insert will also be the first element you take out of the data structure. Queues support two operations: `enqueue()` and `dequeue()`. `Enqueue()` adds an element to the data structure, and `dequeue` removes an element from the data structure.

A queue is normally represented by using a list. However, it can also be implemented by only using Stacks in the internal representation. The problem is: Design a Queue data structure class by only using Stacks (`java.util.Stack` - <http://java.sun.com/j2se/1.4.2/docs/api/java/util/Stack.html>) in the internal representation. Provide a constructor, and support the operations `Enqueue()`, `Dequeue()`, and `Empty()` which returns `True` if the Queue is empty, and `false` otherwise. Be careful to handle the case of underflow errors (when you try to remove an element from the queue when there are no elements in it).

Answers are required to be in Java or C#, but answers will be provided in Java.

Part b.)

In this part, we will ask a relatively simple question. You are given an array of arbitrary length filled with random integers. The elements are in no order. Also, there are duplicates of elements in the array.

Write a function that returns a new array with all the duplicates of any element removed, such that there is only one of each element in the new array.

In this problem, we are looking for an efficient solution. Some aspects of efficiency are space used (number of variables and arrays allocated) and the running time of the algorithm. First try to implement any solution that accomplishes the given task. Then look for ways to improve your solution.

Again, answers are required to be in Java or C#, but solutions will be provided in Java.