**Interview Questions: Stacks and Queues (ungraded)**

**Total points**3

**1.**

**Question 1**

**Queue with two stacks.** Implement a queue with two stacks so that each queue operations takes a constant amortized number of stack operations.

*Note: these interview questions are ungraded and purely for your own enrichment. To get a hint, submit a solution.*

**1 / 1 point**

I don’t understand the question, I will try it later,

**Correct**

*Hint*: If you push elements onto a stack and then pop them all, they appear in reverse order. If you repeat this process, they're now back in order.

**2.**

**Question 2**

**Stack with max.** Create a data structure that efficiently supports the stack operations (push and pop) and also a return-the-maximum operation. Assume the elements are real numbers so that you can compare them.

**1 / 1 point**

we could define a variable named max for our data structure and every time we call the push we compare the value of our operator to the value of the variable that has been added to our data structure, this way when we pop a variable we face some challenges, we don’t now how to replace our max variable if the variable that has been removed was equal to our max variable, so two other solution comes to mind first is that every time we check all the elements of the stack and find the max, second is that we keep a sorted stack along with our first stack. to do every time we want to add a element to our sorted stack we check from the first and put it were the next item is greater and last item is smaller than our input value.

**Correct**

*Hint:* Use two stacks, one to store all of the items and a second stack to store the maximums.

**3.**

**Question 3**

**Java generics.** Explain why Java prohibits generic array creation.

**1 / 1 point**

it is better that we face some compile error when we are trying to compile a code than face a run time error later and the reason is kind of obvious because later that our code is working in some business making a change is difficult, so we use generics and try not two with an exanple feed a human with broken glasses

**Correct**

*Hint:* to start, you need to understand that Java arrays are *covariant* but Java generics are not: that is, \mathtt{String[]}String[] is a subtype of \mathtt{Object[]}Object[], but \mathtt{Stack<String>}Stack<String> is not a subtype of \mathtt{Stack<Object>}Stack<Object>.