CSCI 270 Homework #2

Due Date: Wednesday, February 5th, 11:59pm

1. A stack allows for the following operations. $PUSH(x)$, which adds x to the stack, and POP, which removes the most recent element from the stack. Assume you are given a sequence S of $PUSH$ and POP operations performed on the stack.

   We call a sequence **acceptable** if whenever a POP is called, the stack has at least one element in it. Given S, devise an algorithm to verify if S is valid, and if it is not, describe how to make it valid via adding the minimum possible number of operations to it. You may add an operation at any position in the sequence. As always, explain your answers.

2. Suppose we perform a sequence of n operations on a data structure in which the ith operation costs i if i is an exact power of 2, and 1 otherwise. Analyze the amortized runtime of the operation in question.

3. We can implement a queue using two stacks:

Algorithm 1 Enqueue(x)

```
push(stackB, x)
```

Algorithm 2 Dequeue(x)

```
if stackA is empty then
    while stackB is not empty do
        x = pop(stackB)
        push(stackA, x)
if stackA is not empty then
    return pop(stackA)
```

   You can assume that stack operations $push()$ and $pop()$ take $O(1)$. Analyze the amortized runtime for $Enqueue()$ and $Dequeue()$.

4. You are given a "k-sorted array": this is an almost-sorted array, where each of the elements are misplaced by less than k positions from their correct location. For example, $A = [1, 2, 3, 6, 4, 5, 7]$ is a 3-sorted array, because the elements 4, 5, and 6 are misplaced by 1, 1, 2 positions respectively from their correct locations. Design an $O(n \log k)$ algorithm to sort the array and analyze the runtime. **Hint:** You may find a heap particularly useful.