Mock Online Test: Part B, Question 3

Problem Solving for Computer Science

REMEMBER TO ONLY ATTEMPT TWO OF THE THREE LONG QUESTIONS IN PART B

This question carries a total of 30 marks and consists of sub-questions.

Question 3

This question is about abstract data structures.

1. In the text boxes below, in one sentence explain one difference [2 marks] and one similarity [2 marks] between the queue and the stack abstract data structure.

   One difference:

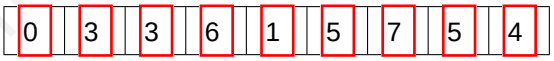
   Stacks are LIFO (last in first out) while queues are FIFO (first in first out)

   One similarity:

   Both structures have an 'empty?' operation that does the same thing.

2. For this next question you will need your nine-digit student number beginning with "03", *not* your campus username. You can find your nine-digit student number on your Goldsmiths ID card, or through the MyGoldsmiths app.

   Once you have your student number, put each digit of your student number into the array so that each element stores one digit.

   Now consider the following sequence of operations:

   - Create a new empty stack
   - Push the first digit of your student number to the stack
   - Push the second digit of your student number to the stack
   - Push the third digit in your student number to the stack
   - Pop the stack
   - Push the fourth digit in your student number to the stack
   - Pop the stack

   (a) In the text box below, write the final digit stored at the top of the stack after these instructions are executed. [2 marks]

   Answer: 3

(b) Either in the text box below or in a separate image file, draw what the stack will look like after the instructions above are executed. Make sure to indicate what are the important elements of the stack. [4 marks]

Since stacks push and pop elements to and from the top of the stack. The element at the bottom of the stack will be 0 with 3 on top of it. The top pointer will point to 3.

If you made an image, put name of the image file here: stack

3. In the file `question3.js` consider the following piece of incomplete JavaScript:

```
1  function Stack() {
2    this.arr = [];
3    this.push = function(item) {
4      if (typeof item === "number") {
5        this.arr.unshift(item);
6      } else {
7        return "Not a number!"
8      }
9    }
10   this.pop = function(item) {
11     if (this.arr.length == 0) {
12       return "Stack underflow";
13     }
14     return this.arr.shift();
15   }
16   this.peek = function() {
17     return MISSING1
18   }
19   this.isEmpty = function() {
20     if (MISSING2) {
21       return true;
22     }
23     return false;
24   }
25 }
```

When completed, this constructor should implement a stack as an object that only stores numbers. In the JavaScript file `question3.js`, complete this constructor by replacing `MISSING1` [3 marks] and `MISSING2` [2 marks] to implement a stack constructor correctly.

4. Go to the file called `question3.js`, and consider the following piece of JavaScript code:

```
1  function stacksEqual(stack1, stack2) {
2    while (!stack1.isEmpty() && !stack2.isEmpty()) {
3      if (stack1 !== stack2) {
4        return false;
5      }
6      stack1.pop();
7      stack2.pop();
8    }
9    return stack1.isEmpty() && stack2.isEmpty();
10 }
```

The function `stacksEqual` is supposed to return a Boolean for two arguments `stack1` and `stack2`, which are both assumed to be stacks containing only numbers. The function should return `true` if both stacks contain exactly the same values in the same order, i.e. the values in the elements are the same; it should return `false` otherwise. However, the function does not work correctly.

(a) In the text box below, explain why the function does not work as supposed, and explain any other problems with the function that you can identify. [7 marks]

From testing I can tell the issue is line 3 in the while statement, it works if you pass the function exactly the same stack twice but it always outputs false if you pass the function two different stacks with the same values in the same order. I imagine you can't directly compare two stacks as a whole. The right way to do it would be to introduce a counter by using a nested for loop in the while loop so that you can compare each value individually. However, these being stacks, we should solve using stack operations. We know that pop! also returns the value poped at the top of the stack. We can use these returned values to check if the values poped from both stacks are equal to each other. Or you could just use peek to read the values of both stacks and compare if they are equal and then pop them if they are. Another improvement that could be made is to check the lengths of both stacks and returning false before performing any operations if they aren't equal lengths. This would save us time instead of getting of getting to the end of both stacks by popping, just to find out both are different lengths. This function skips over the first value in the stack and I have no idea why. Also this whole constructor runs in reverse using unshift rather than just pushing values from the end of the array.

(b) In the text box below, in bullet points or otherwise, briefly explain what alterations to the function `stacksEqual` you can make so that the function now works correctly. [4 marks]

Use the peek operation in the if statement to compare values.
Check if both are stacks are equal lengths before comparing values.

(c) In the file `question3.js`, implement your proposed alterations to the function `stacksEqual`. [4 marks]