Homework 1

Due Friday, May 21, at 11:59 PM ET

Fun Sea Creature Fact: Parrotfish enclose themselves in a bubble of their own mucus at night to prevent predators from smelling them!

Handing In

Homework is handed in through Gradescope. Submit a PDF and match the questions accordingly.

Overview

The following applies to all homeworks:

- We will not accept paper handins. You must submit all written work as a PDF. For information on how to make PDFs of your work, see the very handy **PDF guide**. Please make sure to **NOT** include any identifying information, e.g. name or banner info on your handin, as we are anonymizing grading - we appreciate your help with this.
- Please follow proper pseudocode formatting guidelines. See our **pseudocode standards**. If you use LaTeX, use the `newalg` or `algorithmic` packages or our CS16 `pseudo` environment to format your pseudocode. For even more wizardry, check out the Docs section of our website for a LaTeX handout and tips on improving your pseudocode.

1 Written Problems

Problem 1.1

CS 16 Collaboration, Hours, Tech, and Ed Discussions Policies

If you have not already, take a few minutes to read the **CS16 Collaboration Policy**, the **CS16 Hours Policy**, the **Ed Discussions Policy**, and the **CS16 Tech Policy**.

Once you have finished, fill out **this Google form** regarding the collaboration, hours, and Ed Discussion policy.

You will also need to create an anonymous email for Gradescope setup. Follow this **Google form** for set up.

Next, in the PDF handin for this homework, type "I have read and agree fully to abide by the CS16 Collaboration Policy, the CS16 Hours Policy, and the CS16 Tech Policy." and sign the statement digitally by typing your full name. Note: We realize this is in conflict with the note in the Overview to not include identifying information on your handin. This problem will be the only time in the semester when we request, or allow, you to include it. ***Do not forget to include your signed statement with your homework!***

Problem 1.2

Check out course website + Ed Discussions!

This is really important! Important announcements and all assignments will be posted on the course website, so please make sure to check it out! Also, you must log in to **Ed Discussions** and post on the **welcome thread** by the time that this homework is due. Please introduce yourself, including your favorite sea creature! In addition, in the same post please include a question to either a TA or the staff as a whole. This can be as specific as to 'Where is the PDF that explains SSH?' to as general as 'What is your favorite color?' **Be sure to include your CS login somewhere in the post.**
Note that Ed Discussions collects data of your activities and shares it with companies. You can learn more about the platform and its features in **this link**.

Problem 1.3

How much ethical responsibility does a computer scientist/engineer/designer/etc. have for the technology that they create? Please respond in 4-5 sentences.

Problem 1.4

Write pseudocode for a function, `findConsecutiveMax(array)`, which given an array of characters, returns the character that occurs the greatest number of *consecutive* times

findConsecutiveMax: [array of characters] → character

Examples:

- `findConsecutiveMax([c,c,g,g,h,a,a,a,f,f,a,b,c,c,b,c])` should return `'a'`, because it appears three times in a row – more than any of the other characters. If multiple characters appear consecutively the maximum number of times, your function should return the character whose sequence comes first in the array.

- `findConsecutiveMax([c,a,a,a,b,c,c,c])` returns `'a'`.
- `findConsecutiveMax([s,h,a,k,e,w,e,i,g,h,t])` returns `'s'`.
- If the array is `Empty` or `Null`, return an empty string (`""`).

Problem 1.5

Now, you will write tests for your solution to **Problem 1.4** to check that the program returns the proper output for various inputs. Make sure your tests cover all the edge cases (unusual array length, tie in character counts, etc.). You need at least 4 distinct test cases each with a short explanation to receive full credit. Here is an example test case – you may include a similar (but not the same) test in your answer:

`findConsecutiveMax: [c, a, a, b, b]` $\rightarrow$ `a`

This test case is checking the scenario when multiple characters appear consecutively the maximum number of times. The function returns the first character in the array.

Problem 1.6

Write pseudocode for the following methods of a queue: `enqueue(value)`, and `dequeue()` (which *returns* the element being removed from the queue). The catch is the only data structures you are allowed to use are two stacks, which are stored in instance variables named `s1` and `s2`. **You may not use any other data structures to store your data** (like `arrays` or `ArrayLists`).

You have access to the following stack methods: `push(value)`, `pop()` (which returns the top element), and `isEmpty()` (which returns a boolean). You may call these methods freely on both stacks, and can assume that they will function as per the standard stack definition described below. You can assume that the `pop()` method handles returning an element in the case that the stacks are empty (i.e. calling `pop()` on an empty stack will return null).

Remember, stacks work in a Last In First Out (LIFO) order, meaning the last element that was pushed will be the first element to be popped.

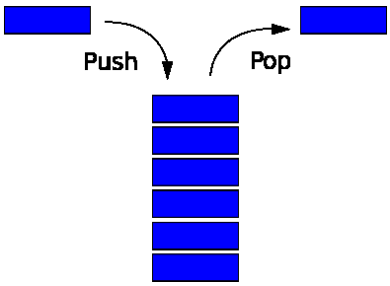

Figure 1: A stack

Conversely, a queue works in First In First Out (FIFO) order, meaning the first element enqueued will be the first element to be dequeued.

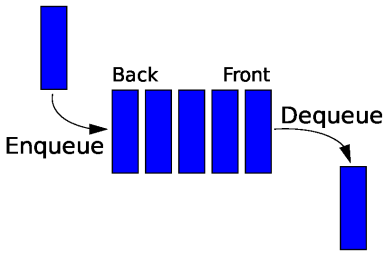

Figure 2: A queue

If you need a refresher on this material, take a look at the lecture slides from CS15 on stacks and queues.

Problem 1.7

Below is a table of calculated importance values. Your task is to find the optimal seam from the top of the table to the bottom. You should provide a sequence of column-indices, each differing from the last by at most 1, with corresponding pixels being the least-total-importance vertical seam. For example, a non-optimal seam [0,1,2,3] would refer to the seam going from top to bottom through col 0, col 1, col 2, and col 3. Hand-simulating this problem will be very helpful in understanding how finding a seam works.

Show your work by turning in the 4×4 cost and directions tables (see slides for a refresher) in addition to the column-indices representing the optimal seam. For the directions table, use 1 to indicate down and to the right, 0 for directly below, and -1 for down and to the left.

2	4	3	1
5	2	1	4
6	3	2	4
1	3	2	1