HW3: Algorithms and Data Structures

On Time Due Date: Sunday November 15th at 2:00 PM

To install the HW3 directory:
    cs015_install hw3
To handin HW3:
    cs015_handin hw3

Getting Started
Install by typing `cs015_install hw3` into the Terminal. This will create a directory in
/home/<your login>/course/cs015/hw3/. This is where you'll need to put the PDF with
your answers. Please include your Banner ID at the top of the PDF, but not your name or login,
as we grade anonymously.

Assignment
1. You are given the string “sharpay” and told to sort the characters alphabetically. Write
   out the sort step-by-step, i.e. write what the string would look like after each
   iteration/recursion of the following algorithms:
      EXAMPLE: Selection sort, where the blue letter is the letter we’re looking at and
      the red letter is the letter we’re swapping with:

      sharpay → ahsrpay
      ahsrpay → aasrphy
      aasrphy → aahrpsy
      aahrpsy → aahprs
      aahprs → aahprs
      aahprs → aahprs

      a. Insertion sort
      b. Merge sort

2. Coach Bolton determines who joins the basketball team by giving each person a
   performance score during tryouts. He chooses a cutoff, and anyone above that score is
   part of the varsity team. He reviews the scores just below the cutoff to determine who
   gets to be an alternate for the varsity team.

   Given a sorted (in increasing order) doubly linked list of performance scores and the
   chosen cutoff value, write an algorithm to return the lowest score chosen as an alternate.
A person will be an alternate if, and only if, their score is within 6 nodes of the cutoff node and if they are also within 3 points of the cutoff score.

Your algorithm must run in $O(n)$ time (where $n$ is the number of nodes in the linked list) and should be written in pseudocode.

**Note:** You can assume the following:
- The cutoff is an element that is contained in the linked list.
- There are at least 6 nodes in the linked list before the cutoff.

If you choose to use `compareTo()`, you may **not** assume it is implemented for you.

```java
// Input: a doubly LinkedList<Double> containing players’
// scores & a double that is the cutoff score.
// Output: lowest grade that is bumped up.
method lastAlternate(scoresList, cutOff):
    /* Your pseudocode goes here */
```

3. It’s been said that if you have at least 23 people in a room, there’s a 50% chance of two people having the same birthday. You’ve been given a list of $n$ birthdays, where $n$ is the number of people in a room. You want to check to see how many duplicate birthdays there are in this list.
   a. First, describe a brute force* solution to this problem. What would be the runtime of this solution?
   b. Now, describe a more efficient solution to this problem that uses a hashset. What would be the runtime of this solution?

   * A “brute force” solution is one that correctly solves a problem in a naive way, but is typically not the most efficient solution to the problem.

   **Note:** We’ll cover hashsets in lecture on Thursday, November 12. You may not be able to fully complete question 3 until then.

4. You are given a stack and a queue. The stack is empty, but the queue is not and contains the numbers 0 through 9 in order, i.e., the head of the queue contains the number 0, while the tail contains the number 9.

   By passing the numbers back and forth between the stack and the queue, reverse the order in the queue of the last $n$ elements. That is, if $n$ equals 4, the final queue should hold numbers in the following order: 0, 1, 2, 3, 4, 5, 9, 8, 7, 6.

   **Note:** You should not assume that $n$ is positive, or that it is less than or equal to 10. If $n$ is negative or if it is larger than the size of the queue, the method should do nothing.
Once again, your answer should be in pseudocode.

// Input: A Stack and a Queue both of size 10, and a
// number n
// Output: A Queue with the last n elements reversed
method reverseLastElements(stack, queue, n):
    /* Your pseudocode goes here */

5. The Fibonacci sequence are the numbers in the following sequence of numbers:
   0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

   By definition, the first two elements in the Fibonacci sequence are either 1 and 1, or 0
   and 1, depending on the chosen starting point of the sequence, and each subsequent
   number is the sum of the previous two. (Note that this sequence is sometimes shown as
   starting at 1. For the purpose of this assignment, you should assume that it starts at 0.)

   Use recursion to output the nth element of the Fibonacci sequence. Fill in the following
   function such that it returns the nth number in the Fibonacci sequence. Assume n > 0.

   Examples:  n = 1, returns 0
               n = 2, returns 1
               n = 5, returns 3
               n = 7, returns 8

   Note: This should be written in pseudocode that very closely resembles Java.

   // Input: element of the fibonacci seq we are looking for
   // Output: The value of the nth number.
   method fibonacci(int n):
       /* Your code goes here */

Handin Info:
This assignment must be submitted no later than Sunday November 15th at 2:00 PM. There is
no late handin for this assignment. You will submit this electronically! Remember to create a
single PDF from your answers and place them in the correct directory (/home/<your
login>/course/cs015/hw3). Then you can run cs015_handin hw3 to submit. Be sure to
include your Banner ID at the top! When you have successfully handed in the assignment, a
confirmation email will be sent to your CS department account (<yourlogin>@cs.brown.edu).