ECS 32B – Introduction to Data Structures
Homework 06

Due: Tuesday, December 1, 2020, 5:00pm PST

Important:

- The purpose of the homework is to practice **hashing and tree traversal**. If you didn't meet the requirements as stated in the description, only half of the points will be given.
- **The homework contains two parts.** The first part does not need any programming. You may writen your answers on paper or type it. Each part has its own submission locker on Gradescope.
- For Part 2, you may upload as many times as you want before the deadline. Each time the autograder will let you know how many of the test cases you have passed/failed. To prevent people from manipulating the autogader, the content of only half of the test cases are visible.
- The testing files contain the test cases for which Gradescope shows the content. You may use the testing files to test your code locally before submitting to Gradescope. However, it does not contain the rest of the test cases, for which Gradescope does not show the content. So passing all the test cases in the testing files does not necessarily mean you will get 100.
- Please **do not copy others' answers**. Autograder can detect similar code.

Part 1: Non-programming (30 points)

1. (2 points) Given a properly implemented hash function *hash*. Suppose two keys x and y are equal, i.e., $x = y$. What is the relationship between their hash codes $hash(x)$ and $hash(y)$?
2. (5 points) Suppose we have a hash table of size 29 and we want to implement a system for looking up a friend's information given his/her name. Which of the following hash functions is the best? Justify your choice. (Assuming none of the names is empty.)
   (a) $h(x) = x$
   (b) $h(x) = n$ % 13, where n is the number of characters in x
   (c) $h(x) = (ord(x[0]) + ord(x[-1]))$ % 29, where $x[0]$ is the first character in the name and $x[-1]$ is the last.
   (d) $h(x) = 29$
   (e) $h(x) = \sum_{i=0}^{n} ord(x[i])$, where n is the number of characters in x, $x[i]$ the ith character in x, and $ord(c)$ returns the ordinal value of a character. (Instruction on the Σ summation symbol `https://www.mathsisfun.com/algebra/sigma-notation.html`)

3. Given a hash table with size 7, a hash function $hash(x) = x \% 7$, and keys 5, 7, 14, 1, 19, 21.
   (a) (5 points) Hash the keys using chaining to resolve collisions. Show your work.
   (b) (5 points) Hash the keys using linear probing to resolve collisions. Show your work.
   (c) (5 points) Hash the keys using quadratic probing to resolve collisions. Show your work.
   (d) (5 points) Hash the keys using double hashing to resolve collisions. Suppose the second hash function $hash'(x) = 3 - (x \% 3)$. Show your work.
   (e) (3 points) What is the load factor after the keys are hashed?

Part 1: Programming (70 points)

1. (25 points) A palindrome is a string that reads the same backward as forward. For example, "madam" and "test tset" are palindromes but "banana" is not. Given a string `s`, write a function `canFormPalindrome(s)` that tests whether the letters in `s` can be permuted to form a palindrome.

   Note:
   - Your algorithm must use hashing (or dictionary in Python).
   - The time complexity is $O(n)$, where n is the length of the input string.
   - Assume an empty string is a palindrome.

   Examples:
   - `canFormPalindrome(aamdm)` should return `True`
     Explanation: "aamdm" can be permuated to "madam", which is a palindrome.
   - `canFormPalindrome(abbddaaa)` should return `True`
     Explanation: "abbddaaa" can be permuated to "aabddbaa", which is a palindrome.
   - `canFormPalindrome(abcc)` should return `False`

2. (25 points) Suppose you need to write an anonymous letter and disguise your handwriting. You decide to cut the characters you need from a book and paste the characters on a paper to form the letter. (Note that we mean a physical book and a physical letter, not digital.) Given the content of a book `book` and the content of the letter `letter` (each is reprented as a string), write a function `anonymousLetter(book, letter)` that determines if it is possible to write the anonymous letter using the book.

   Note:
   - Your algorithm must use hashing (or dictionary in Python).
   - The time complexity is $O(max(m, n))$, where m is the length of the book string and n is the length of the letter.

   Examples:
   - Let the book be "I am a book. Not a letter." and the letter be "I am a letter."
     `anonymousLetter(book, letter)` should return `True`.

- Let the book be "I am a book. Not a letter." and the letter be "I am a letter!" `anonymousLetter(book, letter)` should return `False`.
- Let the book be "abcde" and the letter be "abcdef" `anonymousLetter(book, letter)` should return `False`.

3. (20 points) In class, we discussed the preorder traversal of a binary tree. The code in the slides is recursive. Rewrite the `preorder` function such that
   - it returns the nodes' values in a Python list instead of printing them out, and
   - it is iterative instead of recursive.