Homework 6

DUE: November 6th, 11:59 PM

This assignment covers hash-tables and graphs covered in the class based on zybook actives and content. It is also using the PairProgramming Groups.

1) **Quadratic Probing. [10 pt]**

Assume that c1 = 1 and c2 = 1 and that you are given the following hash table and the hash function is h(key) = key % 15.

0	
1	
2	
3	**3**
4	
5	**5**
6	
7	
8	
9	
10	
11	
12	
13	
14	**14**

Using Quadratic Probing insert 21, 25, 18, 76 and 36, 12 in the given order to the table above (Each correct placement is 1 points):

Search and remove the 36 from the table (1 point).

Show each step in detail. Please refer to search and removal section of 5.4 (participation activity) for an example. That is, if there is collusion occurs, show it is handled, differentiate removal from beginning and empty-after-removal etc. (3 points):

2) Double Hashing [9 pt]

Assume **h1**: **key % 10** and **h2**: **key % 3**

0	
1	
2	
3	
4	
5	**5**
6	
7	**7**
8	
9	
10	**10**
11	
12	
13	
14	

Using double hashing, insert 4, 12, 21, 17, 31 in the given order to the table above (Each correct placement is 1 points):

Please search and remove 17 from the above table (1 points).

Show each step in detail. Please refer to search and removal section of 5.4 (participation activity) for an example. That is, if there is collusion occurs, show it is handled, differentiate removal from beginning and empty-after-removal etc. (3 points).

3) Linear Probing [11 points]

Assume **h(key): key % 15**

0	
1	
2	
3	
4	
5	**5**
6	
7	**7**
8	
9	
10	**10**

11	
12	
13	
14	

Using linear probing, insert 6, 25, and 17. Then search for and remove 10 and then insert 12, 34 in the given order to the table above (Each correct placement is 1 points).

Please search for and remove 25 from the above table (2 points).

Show each step in detail. Please refer to search and removal section of 5.3 (participation activity) for an example. That is, if there is collusion occurs, show it is handled, differentiate removal from beginning and empty-after-removal etc. (3 points).

4) (70 points)

The goal of this assignment is to practice using PUSH and POP operations on a STACK.

Complete the provided Python program by adding code to the

function ***isPalindrome***: a7.py Download a7.py

1. The function ***main*** (in *a7.py*) continually asks the user for a string, and calls *isPalindrome* to check whether each string is a palindrome.
    - A palindrome is a word or sequence that reads the same backward as forward, e.g., *noon*, *madam* or *nurses run*.
    - Your function must ignore spaces and punctuation: when the user enters '*nurses run*', the function returns True, to indicate that it is a palindrome.
2. In your code (in *isPalindrome*) you **must use a stack** to determine whether the input strings are palindromes. The program imports the *Stack* class available

   here: stack.py Download stack.py. Save both files (*stack.py* and *a7.py)* in the same directory, then modify and test *a7.py*.
3. **Do not make any changes to *main()* or to *stack.py*.**

IMPORTANT: You must respect the LIFO property of a stack. Do not use something like: stk.items[-1], or stk.items == stk.items[::-1]. You should not access an item that is at the bottom of the stack without first removing the items on top of it (using pop). Do not use reverse() or similar built-in functions or methods. You may use the methods from the class Stack defined in stack.py.

WHAT TO SUBMIT: submit the modified ***a7.py*** here on Canvas.

The names of the programs are important; please follow the directions carefully, failure to do so will result in a deduction. *If you re-submit the assignment, Canvas will append a*

number to the name of your files. That is okay. You must also include thorough comments for full credit, including your name at the top of the program.

Late policy: up to 2% deduction per hour late (automatically deducted by Canvas).

Rubric:

- Characters from given sentence are pushed onto stack and LIFO respected (15 points)
- Characters are popped from the stack and LIFO respected (15 points)
- Spaces are ignored either by removing them from the original sentence or when pushing/popping to/from stack (15 points)
- Correct Value returned (15 points)
- Main.py and stack.py are unchanged (10 points)

Extra-Credit

Hash Functions and Their Properties [12 pt] (remember the slide IdealHashFunctions, class discussions, and do not hesitate the search over the internet).

a) [2 pt] In-class, we discussed that a keyed hash relies on weaker assumptions than the one without the secret random key for collision-freeness. Explain with a few sentences why we call it "weaker assumption" once a secret random key is introduced into hash (Hint: Remember the discussion about adversarial point of view).

b) [6 pt] *Birthday paradox* refers to the probability that, in a set of *n* randomly chosen people, some pair of them will have the same birthday. It is a simple calculation example (for example 23 people with 365 days), but in essence, via pigeonhole principle (a very intuitive concept, check it if you are not familiar with it over the internet), dictates the probability of collusions in all hash functions (e.g., as indicated in slide 8 for cryptographic hash functions). To simplify the problem, variations (e.g., leap years, twins, seasonal etc…) are omitted, and all 365 possible birthdays are equally likely. We assume there are 23 people in the room. Show in three-four steps, how you could compute the probability *P(A)* which denotes at least two people in the room have the same birthday. Note that you are not required generalize your answer to any number of people, neither use approximations, just simple/plain probability calculation is sufficient. *Hint:* You may want to start first by finding the probability that no two people in the room have the same birthday and take it from there. You can rely on widely used sources on the internet, as long as you cite and express it with your own words and explanations. DO NOT copy-paste answers from the web.

c) [2 pt] Let H be an ideal cryptographic hash function produce a *d-bit* output. Per birthday

paradox concept discussed in the slides, to achieve a collision on two distinct messages with at least probability $\frac{1}{2^{11}}$, how many random messages must be hashed (express it in terms of d-bit, answer can be deduced from slide). Explain your answer with 1-2 sentences.

d) [2 pt] What are the four properties of an ideal (cryptographic) hash function? Each property is 0.5 pt, total 2 pt (see slides).