ELEC 278

Lab 5: Heaps & Hash Tables

Weeks 9 & 10
Fall 2019

Contents

1. Objective 2
2. Organization 2
3. Step 1: Solving a programming problem using heaps 2
   3.1 Task 2
   3.2 Deliverable 3
4. Step 2: Complete Hash Table functions 4
   4.1 Tasks 4
   4.2 Deliverable 4
5. Stretch Goals 4
   5.1 Erase function for hash table. 5
   5.2 Enlarging Table when load factor gets high. 5
   5.3 Using words from a file. 5
ELEC 278 LAB EXERCISE 5 CHECK-OFF SHEET 6

1. Objective

The objectives of this lab are:
a) To practice using Heaps as solution to a programming problem
b) To implement a simple hash table

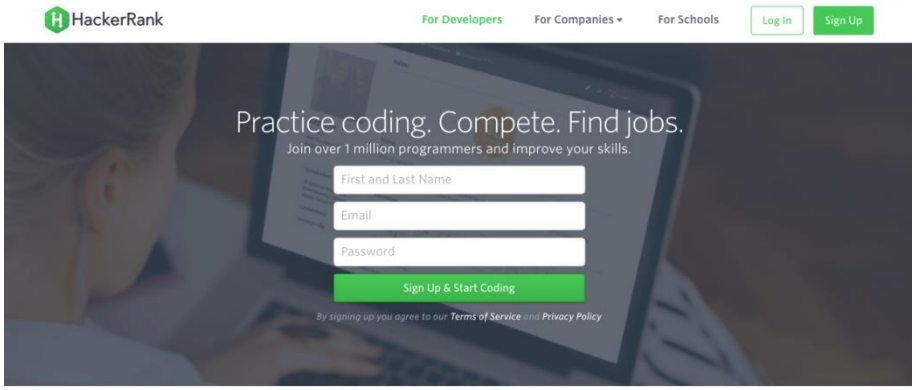
2. Organization

This lab includes two steps:
a) For step 1 (Section 3), you will use heaps to solve a programming problem in Hackerrank.
b) For step 2 (Section 4), you will implement a hash function.
c) For stretch goals (Section 5), you are given several choices. You may implement any one (or more) that you wish.

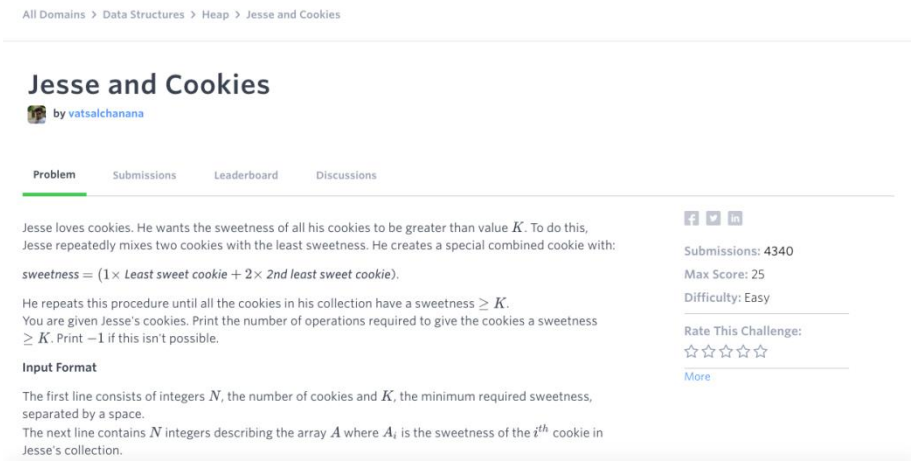
3. Step 1: Solving a programming problem using heaps

3.1 Task

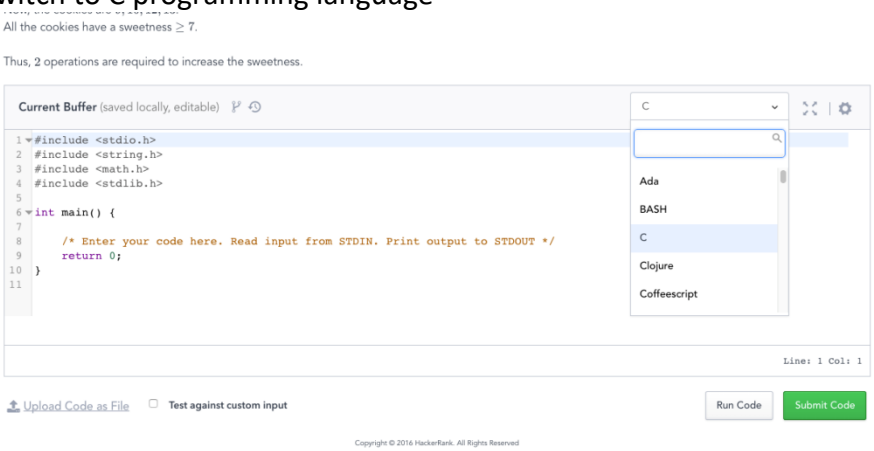
1- You have code for an implementation of minheap in minheap.c (step 1 directory or folder). You will see that there are two versions of main(). One is intended to be used to test your minheap code; the second is used to submit to the hackerrank web page to solve the programming problem. The second has the I/O code supplied, but no operational code after that.
2- Signup and log in in https://www.hackerrank.com/



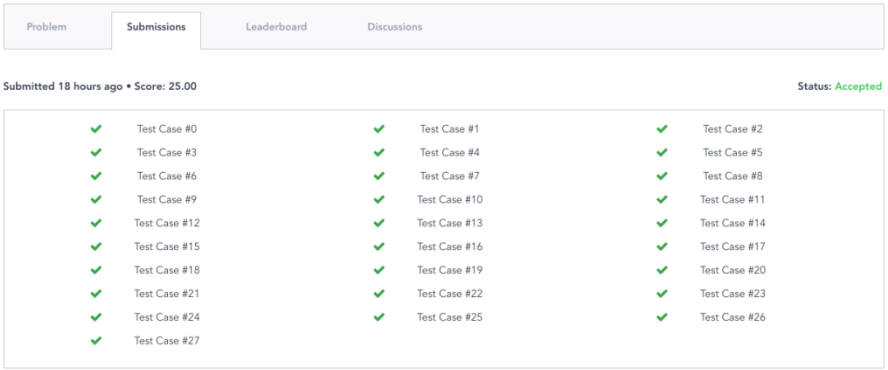
3- Open this challenge: https://www.hackerrank.com/challenges/jesse-and-cookies



4- Switch to C programming language



5- Use the code in minheap.c to solve this challenge.
6- Write your code in the online text editor. (You should be able to just copy and paste.)
7- Run your code to test it against simple test cases.
8- If your code passes the simple test cases, submit it to hacker rank
9- If your code is correct, you will get this page:



3.2 Deliverable

Show your code and the results page to your TA.

4. Step 2: Complete Hash Table functions

In this step you will complete some functions in hash.c file. The hash table is used to store strings using linked lists to resolve collisions. The linked list code is included in DLL.h.

1- Insert function

   Write the code for the insert function that will take a string and find its hashcode (address in table) then inserts it into the linked list of that address. If the list has old value, then a collision is occurred, and the collision counter will be incremented. Use addfront method in DLL.h

2- Find function

   Write the code for the find function that will take a string and find if it is in the table or not. Use find function of the LinkedList.

3- Implement a string hash function

   Write a function to compute the hashcode of a string. Use the following formula:

$$h(s) = \sum_{i=0}^{n-1} S[i].31^{n-1-i}$$

   where S[i] means the character at index i. The term is the ith character multiplied by 31 to the power n-1-i. The main function will create 100 random words (actually just sequences of characters) and insert them into the hash table. Then it prints the table and the number of collisions.

4.1 Tasks

a) Open the file hash.c

b) Complete the functions needed.

c) Compile your code:

   cc hash.c –o hash

d) Run your code and confirm your results.

4.2 Deliverable

Demonstrate your code to the TA and obtain a signature on your sign-off sheet.

5. Stretch Goals

For Lab 5, several stretch goals are offered. You may do 1 or more of them. Remember to get the TA to sign off on the work you do.

5.1 Erase function for hash table.

Implement an erase function – remove_key (x) – that removes a key from the hash table. Check for attempts to remove non-existent keys, handle removal of the last key stored for a particular hash code, and any other special conditions that you can think of. Test your function by removing keys, printing the table after a removal.

5.2 Enlarging Table when load factor gets high.

The load factor is the percentage of the hash table that is occupied. A goal is to keep this number lower than 100% - in fact, 65 -70% might signal a threshold that requires a response. The response would be to increase the size of the table. Modify the code so that the load factor is updated every insert, and when it exceeds 60%, the table is doubled in size. Note that all the entries in the small table need to be placed in the larger table. Do they hash to the same index when the table is twice as large? How does the answer to that question determine how much work needs to be done when growing the table?

5.3 Using words from a file.

A set of words has been provided in a file called words.txt. Modify main() so that, instead of generating random character sequences, the words from this file are used.

ELEC 278 LAB EXERCISE 5 CHECK-OFF SHEET

Print this sheet and bring it to the lab with you. The TA will sign off the steps as you complete them. Make sure you hand in your signed sheet before you leave the lab.

ONE SHEET PER STUDENT PLEASE!

Student Name (optional)	
Student Number (8 digits)	

Part Number	Step Description	Mark	TA Sign Off
Step 1 (Section 3)	Using a heap to solve a programming problem	4	
Step 2 (Section 4)	Complete hash table functions	6	
Stretch (5.1)	Erase function for hash table.	2	
Stretch (5.2)	Enlarging table	2	
Stretch (5.3)	Reading file	1	
	Full marks (complete lab requirements)	10	
	Maximum possible marks	15	