Lab 6 Additional Practice Problems

EECS 281 Fall 2018

Multiple Choice

1. Which of the following statements is **FALSE**?
   - **A)** the `std::map` data structure is implemented using a binary search tree under the hood
   - **B)** the worst-case time complexity of searching a binary search tree occurs when the nodes are in a stick formation
   - **C)** hash tables are extremely useful because searching always takes $\Theta(1)$ time
   - **D)** the dictionary ADT can be implemented using a hash table
   - **E)** none of the above

2. Which of the following statements is **TRUE**?
   - **A)** for a hash table of size 25, index compression maps an index into the range [0, 25]
   - **B)** the `rand()` function, which generates random numbers, is great for hashing since it greatly reduces the chances of collision
   - **C)** both unordered and ordered maps have $\Theta(1)$ average search and insert time complexities
   - **D)** a `std::unordered_map` must be used if you want to keep track of the number of students who have birthdays on each of the 31 days in January
   - **E)** none of the above

3. An easy way to compress an integer key into a hash table of size *M* is to mod it by *M* (i.e. *key* mod *M*). For which of the following collection of keys is this compression method the most ideal?
   - **A)** a list of all scores on a 20-question multiple choice exam, where each question is worth 5 points
   - **B)** a collection of "unluckiest numbers" collected in a survey of 500 Ann Arbor adults
   - **C)** the amount of time Dr. P spends with each student during office hours the day a project is due
   - **D)** the number of credits each full-time student at the university is taking this semester
   - **E)** the collection of student IDs of all students currently enrolled in EECS 281

4. Given two unsorted arrays of distinct numbers, you want to find all pairs of numbers, one from array 1 and one from array 2, that sum to a given value. For instance, if you are given

```
arr1[] = {1, 2, 3, 4, 5, 7, 11};
arr2[] = {2, 3, 4, 5, 6, 8, 12};
```

   you would return `(1,8)`, `(3,6)`, `(4,5)`, `(5,4)`, and `(7,2)`. What is the average time complexity of doing this if you use the most efficient algorithm?
   - **A)** $\Theta(1)$
   - **B)** $\Theta(\log n)$
   - **C)** $\Theta(n)$
   - **D)** $\Theta(n \log n)$
   - **E)** $\Theta(n^2)$

5. Two pairs (a, b) and (c, d) are symmetric if b = c and a = d. Suppose you are given an array of pairs, and you want to find all symmetric pairs in the array. The first element of all pairs is distinct. For instance, if you are given the following array

```
arr1[] = {(14,23), (11,2), (52,83), (49,38), (38,49), (2,11)};
```

you would return `{(11,2), (2,11)}` and `{(49,38), (38,49)}`. What is the average time complexity of doing this if you use the most efficient algorithm?

A) $\Theta(1)$
B) $\Theta(\log n)$
C) $\Theta(n)$
D) $\Theta(n \log n)$
E) $\Theta(n^2)$

Consider the following code for questions 6-7.

```
1   #include <iostream>
2   #include <string>
3   #include <utility>
4   #include <unordered_map>
5   using namespace std;
6
7   int main() {
8      unordered_map<string, string> myMap;
9      myMap.insert(make_pair("Paoletti","Darden"));
10     myMap.insert(make_pair("Almomani","Darden"));
11     myMap.insert(make_pair("Paoletti","Almomani"));
12     myMap["Almomani"] = "Paoletti";
13     myMap.insert(make_pair("Paoletti", "Markov"));
14     cout << myMap["Paoletti"] << endl;
15     cout << myMap["Darden"] << endl;
16     myMap.erase("Paoletti");
17     cout << myMap["Almomani"] << endl;
18     cout << myMap.size() << endl;
19  } // main()
```

6. What does line 17 print?
   A) Paoletti
   B) Darden
   C) Almomani
   D) Markov
   E) an empty string

7. What does line 18 print?
   A) 1
   B) 2
   C) 3
   D) 4
   E) 7

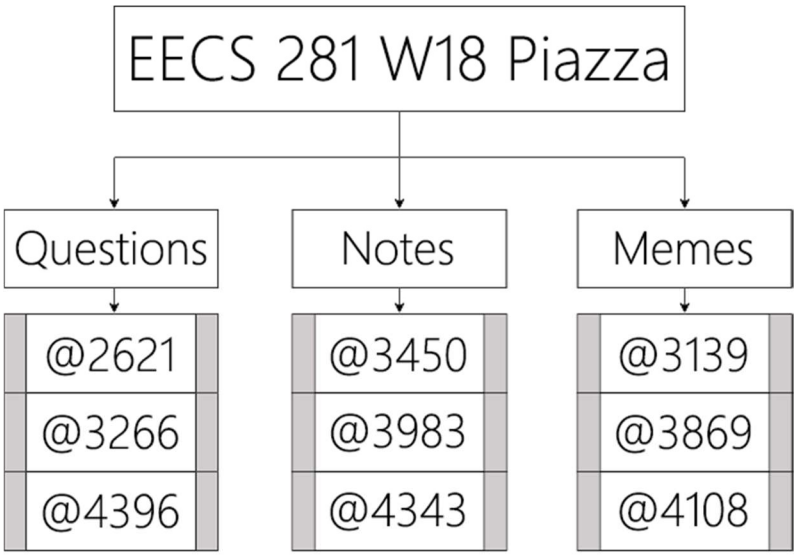
8. The Project 3 autograder just got released! On the first day, eight students submitted. The scores that each of these students received on their first submission are shown in the table below.

Name	Alice	Bob	Cathy	Drew	Erin	Frank	Grace	Henry
Score	34.9	46.0	14.6	65.1	42.1	28.0	96.7	74.5

Suppose these students are placed into a `std::map` named `P3Scores`, where *Name* represents the key and *Score* represents the value. If `it = P3Scores.end()`, what is the value of `(--it)->first`?

A) Drew
B) Cathy
C) Frank
D) Grace
E) none of the above

9. Suppose that, on the Winter 2018 EECS 281 Piazza page, posts can be categorized into three unique groups: questions, notes, and memes. A representation of this is shown below:



Suppose you implemented the following unordered map that maps the type of each post to the post IDs that correspond to each of these types (e.g. the key `notes` maps to `{3450,3983,4343}`):

```
std::unordered_map<std::string, std::vector<int>> posts;
```

While searching through Piazza, you discover that post @4753 is a meme. Which of the following successfully appends 4753 to the vector associated with the key `memes`?

A) `posts["memes"] = 4753;`
B) `posts["memes"] = memes.push_back(4753);`
C) `posts["memes"] = posts.push_back(4753);`
D) `posts["memes"].push_back(4753);`
E) `posts["memes"].second.push_back(4753);`

10. You have a spreadsheet with post data from the Fall 2018 EECS 281 Piazza site. The data you have is shown in the table below:

Post ID	Post Author	View Count	Posted by Instructor	Subject Line
509	Oliver Hill	490	True	Potatobot lives!
1595	Joseph Nwabueze	373	True	Reducing Memory Usage of Classes
3162	Jeremy	89	False	HOW ARE CHILDREN MADE
3471	Anonymous	90	False	Pizza announcement?
3882	Anonymous	339	False	nice
3959	Andrew Zhou	522	True	Written Scores Released on Gradescope!

Suppose that you create a `std::map` with post ID as the key and subject line as the value. You insert the six Piazza posts listed above into the map and run the following code:

```
1    std::map<int, std::string> piazza;
2    piazza[509] = "Potatobot lives!";
3    piazza[1595] = "Reducing Memory Usage of Classes";
4    piazza[3162] = "HOW ARE CHILDREN MADE";
5    piazza[3471] = "Pizza announcement?";
6    piazza[3882] = "nice";
7    piazza[3959] = "Written Scores Released on Gradescope!";
8    auto it = piazza.lower_bound(3000);
9    std::cout << (it++)->second;
```

What gets printed?

A) `Potatobot lives!`
B) `Reducing Memory Usage of Classes`
C) `HOW ARE CHILDREN MADE`
D) `Written Scores Released on Gradescope!`
E) none of the above

11. Suppose you ran the following code instead:

```
1    std::map<int, std::string> piazza;
2    piazza[509] = "Potatobot lives!";
3    piazza[1595] = "Reducing Memory Usage of Classes";
4    piazza[3162] = "HOW ARE CHILDREN MADE";
5    piazza[3471] = "Pizza announcement?";
6    piazza[3882] = "nice";
7    piazza[3959] = "Written Scores Released on Gradescope!";
8    auto rev_it = piazza.rbegin();
9    std::cout << (++rev_it)->second << " ";
10   for (int i = 0; i < 3; ++i) ++rev_it;
11   std::cout << (rev_it--)->first << " ";
12   std::cout << (--rev_it)->first << " ";
13   std::cout << (--rev_it)->second << " ";
```

What gets printed now? nice 1595 3471 nice

On Piazza, you can retrieve live statistics of the top five student contributors on the site! The following statistics were retrieved from the Fall 2018 EECS 281 Piazza page on November 3, 2018. Suppose that these statistics were gathered and inserted into a hash table that maps a student's name to the number of Piazza contributions they've made, as shown by the code below:

```
1    // 281 Piazza Post Statistics, 3 November 2018
2    // Map from uniqname to pair<first_name, num_contributions>
3
4    #include <iostream>
5    #include <string>
6    #include <vector>
7    #include <utility>
8    #include <map>
9    #include <unordered_map>
10   using namespace std;
11
12   int main() {
13      unordered_map<string, pair<string, int>> contributors;
14      contributors["andersej"] = make_pair("Jeremy", 478);
15      contributors["abosch"] = make_pair("Alex", 345);
16      contributors["akiek"] = make_pair("Austin", 333);
17      contributors["eashwar"] = make_pair("Eashwar", 323);
18      contributors["philiyao"] = make_pair("Philip", 229);
19
20      vector<string> vec;
21      for (auto myVar : contributors) {
22         vec.push_back(myVar.first);
23      }
24
25      cout << vec.front() << endl;
26
27      return 0;
28   } // main()
```

Use the above code to answer questions 12-15.

12. For which of the following operations would a `std::unordered_map` **NOT** be ideal?
    - A) identifying the student who has the most Piazza contributions
    - B) identifying the first name of the student with the uniqname `andersej`
    - C) identifying the number of times `abosch` has answered a Piazza post
    - D) updating `akiek`'s post count whenever he answers a question
    - E) all of the above procedures are ideal using a `std::unordered_map`

13. What is the type of `myVar` on line 21?
    - A) `string`
    - B) `pair<string, int>`
    - C) `pair<string, string>`
    - D) `pair<string, pair<string, int>>`
    - E) `pair<pair<string, int>, string>`

14. What does line 25 print?
    A) `andersej`
    B) `Jeremy`
    C) `abosch`
    D) `Alex`
    E) impossible to determine

15. Suppose that line 13 were replaced with the following line:

```
map<string, pair<string, int>> contributors;
```

    What does line 25 print now?
    A) `andersej`
    B) `Jeremy`
    C) `abosch`
    D) `Alex`
    E) impossible to determine

Written Practice

16. Users on longer flights like to start a second movie right when their first one ends, but they complain that the plane usually lands before they can see the ending. You want to develop a method for choosing two movies whose total runtimes will equal the exact flight length.

    Write a function that takes an integer `flightLength` (in minutes) and a vector of integers `movieLengths` (in minutes) and returns a boolean indicating whether there are two numbers in `movieLengths` whose sum equals `flightLength`. The passengers do not want to watch the same movie twice. Your solution should have a runtime complexity of $\Theta(n)$, where n is the length of `movieLengths`, and it should be done in at most 15 lines of code.

```
bool identifyMovies(const vector<int>& movieLengths, int flightLength) {

  unordered_set<int> movieLengthsSeen;
  for (int firstMovieLength : movieLengths) {
     int secondMovieLength = flightLength – firstMovieLength;
     if (movieLengthsSeen.find(secondMovieLength) != movieLengthsSeen.end())
       return true;
     movieLengthsSeen.insert(firstMovieLength);
  }
  return false;

}
```