CS 284 C: Quiz 3

Spring 2020

Time: 15 minutes

Student Name:
Honor Pledge:

1. (big-O, 1pt). Derive the big-O notation for the following code. You must provide details on how it was established. You may assume that $n > 1$.

```java
for(int i=1; i<n; i++) {
    for(int j=1; j<n; j*=2) {
        System.out.println(i + " " + j);
    }
}
```

2. (big-O, 4 pts). Recall that in class, we proved the soundness and completeness of using the two pointers algorithm for solving the two-sum problem. We say the two-pointers algorithm is *sound*, because every solution it outputs is a correct solution; we say the two-pointers algorithm is *complete*, because whenever there exists a non-empty solution, the algorithm can guarantee to find the solution.

(1) (weighted two sum, 2 pts). Recall that we used the following matrix to prove that the two-pointers algorithm is complete for the two sum problem:

```java
    int[] nums = {1, 2, 4, 8, 16, 32};
    int target = 12;
```

$$\begin{array}{c|ccccc|}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
0 & 3 & 5 & 9 & 17 & 33 \\
1 & & 6 & 10 & 18 & 34 \\
2 & & & 12 & 20 & 36 \\
3 & & & & 24 & 40 \\
4 & & & & & 48 \\
\hline
\end{array}$$

We will work on a slightly different problem than two sum. Suppose that we are given a sorted (ascending) array of unique positive integers, e.g., $nums = \{1,\ 2,\ 4,\ 8,\ 16,\ 32\}$, and a target positive integer, e.g., $target$ =10. The goal is to find two different numbers a and $b \in nums$, such that $a < b$ and $2 * a + b = target$.

We call this problem the *weighted two sum* problem.

By following the same method that we learned in class, can we prove the two-pointers algorithm is also complete for the weighted two sum problem (1 pt)? Can you provide the proof by drawing the matrix for the following example: *nums* = {1, 2, 4, 8, 16, 32} and *target* =10 (1pt)?

(2) (two minius, 2 pts). Now suppose we have a different goal, which is to find two different numbers such that $b - a = target$, where $b > a$. We call this problem the *two minus* problem. Can we still use exactly the *same* matrix method to prove that the two pointers algorithm is *complete* for the two minus problem (1 pt)? Why/why not (1 pt)? (Hint: try drawing the matrix below).

3. (Java basic, 5 pts). In this problem, we will build a database that contains students from Stevens, so that we can search for students by their first names. Every person has a first name, and every stevens student has both a first name and a CWID. First, let's implement an abstract class `Person`:

```java
public abstract class Person {

  private String first_name = "";

  public Person(String first_name) {
    this.first_name = first_name;
  }

  public String get_firstname() {
    return this.first_name;
  }

  /**
   * set the first name of a person
   * @param first_name: the first name of the person
   */
  public void set_firstname(String first_name) {



  }








}
```

(1) (1 pt). Implement the method set_firstname above.

(2) (1 pt). The following class Stevens_student extends the abstract class Person. Implement its constructor method.

```
public class Stevens_student extends Person{

  private int CWID;

  /**
   * Constructor method for Stevens_student
   * @param first_name
   * @param CWID
   */
  public Stevens_student(String first_name, int CWID) {




  }










}
```

(3) (1 pt). Implement the method set_cwid by filling in the blank space of either Person or Stevens_student. Where should you put this method?

(4) (2 pt). Now we can define the database class StevensDatabase. The class StevensDatabase has an array students where each element is an object of type Stevens_student. Implement the method search_cwid for searching a student by his/her first name. The input of search_cwid is a String object target_firstname, which is the first name of the student being searched; search_cwid returns the CWID of the target student if target_firstname exists in the database, otherwise, the method returns -1.

```
public class StevensDatabase {

  private Stevens_student[] students;

  public StevensDatabase(Stevens_student[] students) {
    this.students = students;
```

```java
  }

  /**
   * search for a student's CWID using the student's first name
   * assume there does not exists two students with the same first name
   * @param target_firstname: the target student's first name
   * @return if target_firstname exists in self.students, return the CWID of
   *  that student; otherwise, return -1
   */
  public int search_cwid(String target_firstname) {






  }

}
```