Santa Monica College CS 20A: Data Structures with C++ Assignment 5
Spring 2018

Name: **ID:**

Problem 1:
For all of the following, determine the **total complexity** and then the **Big-O** of the given code segments:

a.

```
for (int j = 0; j < n; j++)
   for (int k = 0; k < j; k++)
         sum++;
```

b.

```
for (int i = 0; i < q*q; i++)
      for (int j = 0; j < i; j++)
            sum++;
```

c.

```
for (int i = 0; i < n; i++)
      for (int j = 0; j < i*i; j++)
            for (int k = 0; k < j; k++)
                  sum++;
```

d.

```
for (int i = 0; i < p; i++)
      for (int j = 0; j < i*i; j++)
            for (int k = 0; k < i; k++)
                  sum++;
```

Name: **ID:**

e.

```
for (int i = 0; i < n; i++)
{
        Circ arr[n];

        arr[i].setRadius(i);
}
```

f.

```
for (int i = 0; i < n; i++)
{
        int k = i;
        while (k > 1)
        {
                sum++;
                k = k / 2;
        }
}
```

Problem 2:
Given a vector of sets of ints, `vector< set<int> > v`, assume the vector v has **N** total sets and that each set has an average of **Q** items.

a. What is the Big-O of determining if the first set, v[0], contains the value 7?

b. What is the Big-O of determining if any set in v has the value 7?

Name: **ID:**

c. What is the Big-O of determining the number of even values in all of v?

d. What is the Big-O of finding the first set with a value of 7 and then counting the number of even values in that set?

Problem 3:
Determine the data structure needed if we wanted to maintain a bunch of peoples' names and for each person, allows us to easily get all of the streets they lived on. Assume there are P total people and each person has lived on average E former streets.

What is the Big-O cost of:

a. Finding the names of all people who lived on "Levering Street"?

Name: **ID:**

b. Determining if “Bill” ever lived on “Westwood Blvd”?

c. Printing out every name along with each person’s street addresses in alphabetical order?

d. Printing out all the streets that “Tala” has lived on?

Name: **ID:**

Problem 4:
Fibonacci numbers are a series of numbers given by the relationship:

$$F_n = F_{n-1} + F_{n-2}$$

With $F_0 = 0 \ and \ F_1 = 1$. In other words, the nth Fibonacci number is given by the sum of the two Fibonacci numbers before it. For Example, the first 13 Fibonacci numbers are:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

a. Implement a recursive function to compute the nth Fibonacci number:

```
void fibonacci(int n) {













}
```

b. What is the Big-O of the recursive Fibonacci function?

Name: **ID:**

Problem 5:

Given the following array show the result after one round of the each of the sorting algorithms indicated. One round being one full iteration of the algorithm's outer most for/while loop.

a. Selection Sort:

99	16	3	19	13	0	13	12	6

b. Insertion Sort:

99	16	3	19	13	0	13	12	6

c. Bubble Sort

99	16	3	19	13	0	13	12	6