Name: [redacted] *(as it would appear on official course roster)*	
Umail address: [redacted] @umail.ucsb.edu	section [redacted]
Optional: name you wish to be called if different from name above.	
Optional: name of "homework buddy" (leaving this blank signifies "I worked alone"	

1

h06

CS24 W19

h06: Chapter 1: Runing Time Analysis

ready?	assigned	due	points
true	Wed 02/13 02:00PM	Mon 02/25 09:00AM	50

You may collaborate on this homework with AT MOST one person, an optional "homework buddy".

MAY ONLY BE TURNED IN IN THE LECTURE/LAB LISTED ABOVE AS THE DUE DATE, OR IF APPLICABLE, SUBMITTED ON GRADESCOPE. There is NO MAKEUP for missed assignments; in place of that, we drop the lowest scores (if you have zeros, those are the lowest scores.)

Complete your reading of Chapter 1, Section 1.2. If you don't have a copy of the textbook yet, there is one on reserve at the library under "COMP000-STAFF - Permanent Reserve").

Please:

- **No Staples.**
- **No Paperclips.**
- **No folded down corners.**

1. (10 pts) What is the Big-O time complexity of the following code:

```
for(int i=0; i<N; i+=2) {
        ...constant time operations...
}
```

Justify your answer.

big-O (N), because the const-time operation is done N/2 times

2. (10 pts) What is the Big-O time complexity of the following code:

```
for(int i=1; i<N; i*=2) {
        ...constant time operations...
}
```

Justify your answer.

big-O (logN), because the const-time operation is done $\log_2 N$ times

3. (20 pts) What is the Big-O time complexity of the following code:

```
int x = 10;
for(int i=1; i<N; i*=2) {
        for(int j=0; j<N; j+=2) {
                x++;
        }
}
```

big-O (nlogn), because x++ is in the for loop with Big-O(n) complexity which is nested in a loop with a Big-O (logn) complexity

4. (10 pts) Provide a link to your github repo for pa02. Analyze the Big-O complexity of the insert, delete and search methods of your implementation of the CardList (linked list class). Assume that Alice and Bob each have exactly N cards. Provide your analysis here. As a challenge problem, analyze the Big-O complexity of your entire game.

2

h06

CS24 W19

Refer to code in the repo h06-q4 in our class organization

- insert:

have a loadCards() method in Player's class, which loads 1 card with a Big-O(1) complexity (inserts to the end, no loop), and loads in N cards with a Big-O(N) complexity.

- delete:

(search method)

a card is found with a Big-O(N) complexity, and deleted with a Big-O(1) complexity. Total complexity: Big-O(N)

- search:

a card is found with a Big-O(N) complexity, because of the sequential search that is performed. In the worst case, it will go through all elements, with total complexity being big-O(n).

- entire game:

since the game goes through the lists of cards of two players and finds matches using the search method, its complexity is Big-O(n^2).