CS2210 Assignment #1

1.) Use the definition of "big oh" to prove $\frac{n+2}{n}$ is $O(1)$.

To prove that $\frac{n+2}{n}$ is $O(1)$ we need to show that there is a real constant $c>0$ and an integer constant $n_0 \geq 1$ such that $\frac{n+2}{n} \leq c \cdot (1)$, for all $n \geq n_0$

Lets first simplify the inequality,

$\frac{n+2}{n} \leq c$, for all $n \geq n_0$

$\Rightarrow n+2 \leq nc$, for all $n \geq n_0$

$\Rightarrow 2 \leq nc - n$, for all $n \geq n_0$

$\Rightarrow 2 \leq n(c-1)$, for all $n \geq n_0$

Now, lets choose $c=2$ to get

$2 \leq n(2-1)$, for all $n \geq n_0$

$\Rightarrow 2 \leq n$, for all $n \geq n_0$

The inequality is valid for all values of n that are atleast 2, so we can choose, $n_0 = 2$.

∴ Since we have found constant values $c=5$, $n_0 = 3$ that make the inequality true, we have proven $\frac{n+2}{n}$ is $O(1)$.

2.) Let $f(n) > 0$ and $g(n) > 0$ for all n and $\frac{1}{f(n)}$ is $O(\frac{1}{g(n)})$.

Prove that $2g(n)$ is $O(f(n))$.

∵ $\frac{1}{f(n)}$ is $O(\frac{1}{g(n)}) \rightarrow c' > 0$, $n_0' \geq 1$ s.t. $\frac{1}{f(n)} \leq c' \frac{1}{g(n)}$ for all $n \geq n_0'$

We must use this knowledge to prove that there is a real constant $c>0$ and an integer constant $n_0 \geq 1$ such that $2g(n) \leq c \cdot f(n)$ for all $n \geq n_0$.

First we will start with,

$\frac{1}{f(n)} \leq c' \cdot \frac{1}{g(n)}$, for all $n \geq n_0'$

$\Rightarrow g(n) \leq c' f(n)$, for all $n \geq n_0'$

$\Rightarrow 2g(n) \leq 2c' f(n)$, for all $n \geq n_0'$

→ Cont.

2.) ... Cont

Lets choose $c = 2c'$ and $n_0 = n'_0$ for all $n \geq n_0$

Now it becomes,

$2g(n) \leq c f(n)$. for all $n \geq n_0$

∴ Since we have found constant values $c = 2c'$, $n_0 = n'_0$, we have proven $2g(n)$ is $O(f(n))$

3.) Use the definition of "big Oh" to prove that $f(n) = \sqrt{n}$ is not $O(\frac{1}{\sqrt{n}})$.

Lets assume that there is some constant $c > 0$ and integer $n_0 \geq 1$ such that

$\sqrt{n} \leq c \cdot \frac{1}{\sqrt{n}}$, for all $n \geq n_0$.

Lets try to derive a contradiction from the inequality.

$\Rightarrow \sqrt{n} \leq c \cdot \frac{1}{\sqrt{n}}$, for all $n \geq n_0$

$\Rightarrow \sqrt{n} \cdot \sqrt{n} \leq c \cdot \frac{1}{\sqrt{n}} \cdot \sqrt{n}$, for all $n \geq n_0$

$\Rightarrow n \leq c \cdot 1$, for all $n \geq n_0$

$= n \leq c$, for all $n \geq n_0$

∴ The inequality $n \leq c$ is only true for values of n that are at most c, so this inequality cannot be true for all values n larger than some constant n_0. Specifically, if we choose $n \geq c + n_0$, then you can see that these values of n are larger than or equal than n_0 but they are not at most c.

∴ we have reached a contradiction as there are no constant values $c > 0$ and $n_0 \geq 1$ s.t. $\sqrt{n} \leq c \cdot \frac{1}{\sqrt{n}}$ for all $n \geq n_0$.

4.) ii) The algorithm does not always terminate. Consider an array of length 2 and x is not inside the array. Since the recursive calls are made before checking if x=L[mid] or first > last, the algorithm will keep updating mid, and will be stuck in an infinite loop of recursion calls.

Specifically Consider L=[3,5], x=1, first=0, last=1

First,

$mid = \left\lfloor \frac{0+1}{2} \right\rfloor = 0$

Then it makes the recursive call with first = 0, last = 0-1 = -1, x=1 and first = 0+1 = 1, last = 1, x=1

Then we have,

binSearch(L, 0, -1, x)

$mid = \left\lfloor \frac{0-1}{2} \right\rfloor = -1$

binSearch(L, 0, -2, x) binSearch(L, 0, -1, x)

binSearch(L, 1, 1, x)

$mid = \left\lfloor \frac{1+1}{2} \right\rfloor = 1$

binSearch(L, 1, 0, x) binSearch(L, 2, 1, x)

Specifically here we see binSearch(L, 0, -1, x) called again from binSearch(L, 0, -1, x). This means the program will be stuck in an infinite recursion and never terminate.

5.) ii) The algorithm is not always correct. Consider the specific case where L is an array of length 1. So, first = last = 0. Also, x = L[0]. The first condition is if first = last return -1. Immediately the algorithm will return -1 and terminate. Since it does not correctly return the index = 0 where x is located, the algorithm is incorrect.

6.)

```
        i=0
        j=0      } c1
10  1   While i+j<n {
    2       if (i>j) {
                i=1 }
            else {
    3           j=j-n }
    4       i=n+i
9   5       While j<i {
    6           if j>0 {
                    A[j+1]=i }
                else {
    7               j=-j
    8               A[j-1]=j
                }
            }
        }
```

$c_2 = 10$

# iter	variables
	i=0 j=0
1	j=-n, i=n, j=n
2	Terminates

$f(n) = 2 + 10 = 12$

$\therefore f(n)$ is $O(1)$

To compute the time complexity, I traced the algorithm while keeping track of the variable updates. As you can see, the outer while loop only completes one iteration because the i+j<n condition is only met once. I then counted the # of operations used within since it is a known constant. We can then add this to $c_1 = 2$ to calculate the relative time complexity.

Proof $f(n) = 12$ is $O(1)$

$12 \le c \cdot 1$ for all $n \ge n_0$

choose $c = 12$ and $n_0 = 1$.

7.)

Linear Search	n
875 ns	5
333 ns	10
834 ns	100
6833 ns	1000
65417 ns	10000
679166 ns	100000

n	Quadratic Search
5	1708 ns
10	1334 ns
100	78875 ns
1000	4506084 ns
10000	17512417 ns

n	Factorial Search
7	11167 ns
8	2125 ns
9	1500 ns
10	1208 ns
11	1292 ns
12	1458 ns