Homework 1 - CptS 223

1. $\frac{2}{N} < 37 < \sqrt{N} < N < N\log(\log(N)) < N\log N < N\log^2 N$
$< N^{1.5} < N^2 < N^2\log N < N^4 < 2^{(n/2)} < 2^N$

2. $O(N) \rightarrow \left(\frac{35}{20}\right)100 = \boxed{175}$
$O(N+\log N) \rightarrow \left(\frac{35}{20\log 20}\right)100\log 100 = \boxed{455.2}$
$O(N^3) \rightarrow \left(\frac{35}{20^3}\right)100^3 = \boxed{4375}$
$O(2^N) \rightarrow \left(\frac{35}{2^{20}}\right)2^{100} = \boxed{42 \cdot 10^{24}}$

(There is no #3)

4. a. $f(n) = f(n-1) + 3 \rightarrow \boxed{O(n)}$
$g(n) = 1 + 2(n+1) + 2n + 1 = 4n+4 \rightarrow \boxed{O(n)}$

b. max depth = 5, f(5)
$\boxed{f(n) = O(n) \; ; \; g(n) = O(1)}$

c.
```
int h(int n)
  {
    if (n<=1)
    {
      return n;
    }
    else
    {
      return h(n-1) + h(n-2)
    }
```
time complexity =
$h(n) = O(1)$

5. runs $\frac{n}{2}$, if n is odd $\rightarrow \frac{n}{2}+1$
if n is even $\rightarrow \frac{n}{2}$
$\Big\} \boxed{O(\frac{n}{2})}$

7. $T(n) = O(n)$

6. 1. Input n.
   2. Declare variables:
      a=1, b=2, c=3, d=4, e=5, f=6, g=7, h=8, i=9, x=1
      count = 0, K = 1;
   3. Get remainder of product.
   4. Do-while loop runs.
   5. Check if remainder is equal to integer and add 1 to count.
   6. Repeat steps 3-5 until count = 10.
   7. Display K.

```
while ( x != 0 )
{
    int p = K* n;
    int r = p % 10;
    do {
        if (r == a)
        {
            a--;
            if(a == -1) { K++;}
        }
        if (r == b)
        {
            b = b-r;
            if (b == 0) { K++;}
        }
        :
        :  // repeats above until i
        :
        if (K == 10)
        {
            x = 0;
            // print K
        }
        p = p/10;
        r = r % 10;
    } while (r != 0);
```

7. a.

```
1. input n
2. check if n%2 ==0. If so, n is even.
3. else, n is odd.
```

$T(n) = O(1)$

b.

```
1. input n
2. for i=0 to lenght
    3. if (n = element at i)
        4. element is found
    5. else, element was not found
```

$T(n) = O(n)$

c.

```
1. s = list[0]
2. for i=0 to lenght
    3. if (s > list[i])
        4. s = list[i]
5. once loop is done, return s.
```

$T(n) = O(n)$

e.

```
1. check lenght of list 1
2. repeat 1 for list 2
3. if ( lenght1 == lenght2 )
    4. for i=0 to lenght1
        5. if (a[i] == b[i])
    6. if all are equal, then lists are the same
    7. else not.
8. else, lenght of list is not equal
```

$T(n) = O(n)$

d. same as e, without steps 1, 2, 3, and 8

$T(n) = O(n^2)$

e.
```
1. node search( root , n)
2.     if root == NULL or root → data == n,  return root.
3.   if root → data < n  ,  return search (root → right, n)
4.    return search (root → left, n)
```

8.

cp	→ copy files / directories
rm	→ removes ...
mkdir	→ make directory
ssh	→ provide secure, encrypted connection
g++	→ GNU C++ compiler
scp	→ copy / transfer files across hosts

9. argc → number of arguments passed (non-neg)
argv [] → points to each argument passed to the program
Both are command line arguments