(Jake Tantorski

Assignment 2

Section 1

1.

```
int i = 1;  //1
while (i <= n)  //
        some O(1) time statements     //1
        i = i*2; //2
end while
```

Iteration 1: i = 1

Iteration 2: i = 2

Iteration 3: i = 4

Iteration k: i = 2^(k-1) = n

k-1=log(n)

K = log(n)+1

Time: O(logn)

2.

Outer loop: n/2^k-1= 1

Using algebra I multiplied each side by 2^k-1.

N = 2^k-1

Logged each side

Log(n) = k-1

Moved the one over

Log(n)+1=k

Middle Loop:

2^k = 1

Logged each side

n-= log(n)

Move the one over

n = log(n)

Inisde Loop:

n/2 = 1

Time: (log(n)+1)*(log(n))*(n/2) = (nlog(n)n)/2 + (nlog(n))/2 = O(nlogn)

3.

Outside Loop: n times

Middle Loop: 2^(k-1) -1 = n --> (k-1) =log(n)+1 --> k = log(n)

Inside Loop: j times = n

Time: n*log(n)*n = n^2*log(n) = O(n^2)

4.

```
j = 1, i = 0; //2

while (i < n) {

            i = i + j;  //2 * Sqrt(2n)

            j++;   //2 * Sqrt(2n)

}
```

Time: 2+4Sqrt(2n) = O(sqrt(n))

Iteration k: (k*(k+1))/2

n = (k*(k+1))/2

2n = k*(k+1)

k = k+1assuming n is large

2n = k^2

k = Sqrt(2n)

5.

2010

log(log(n))

log(n)

Sum of n where i = 1 (1/i)

sqrt(n)

n

Sum of n where i = 1 (1)

nlog(n)

Sum of n where i = 1 (i)

n^2

n^4

2^n

e^n

n!

n^n

6.

```
squareroot(x)

Input X             //2

Lower = 1           //2

Upper = x           //2

Ans = 0             //2

while(lower<=upper)     //logn

        Mid = (lower+upper)/2

        if(mid*mid == x)        //2

                Return mid

        Else if(mid*mid < x)    //2

                lower = mid + 1

                ans = mid

        Else

                Upper = mid - 1

        End if

  End while

Return ans
```

Time: O(log(n)) because it only uses binary search