CSE 222 HW1

1)

a) $\lim_{n\to\infty} \frac{n^2+7n}{n^3+7} = 0 \longrightarrow f(n) = O(g(n))$

b) $\lim_{n\to\infty} \frac{12n+\log_2 n^2}{n^2+6n} = \frac{12n+2\log_2 n}{n^2+6n} = 0 \longrightarrow f(n) = O(g(n))$

c) $\lim_{n\to\infty} \frac{n\log_2 3n}{n+3\log_2 2n} = \infty \longrightarrow f(n) = \Omega(g(n))$

d) $\lim_{n\to\infty} \frac{n^n+5n}{3.2^n} = \infty \longrightarrow f(n) = \Omega(g(n))$

e) $\lim_{n\to\infty} \frac{\sqrt[3]{2n}}{\sqrt{2n}} = \frac{\sqrt[3]{2}\, n^{\frac{1}{3}}}{\sqrt{2}\, n^{\frac{1}{2}}} = 0 \longrightarrow f(n) = O(g(n))$

2) a) For loop inside methodA executes "n" times.
Then worst time complexity is $O(n)$.

b) For loop inside method B executes "n" times.
And it calls methodA. Then worst time complexity is $O(n^2)$.

c) For methodC if n is 0 then while loop cannot be executed and worst time complexity is $O(1)$.
If n is not 0 then while loop goes infinite loop and worst time complexity cannot be calculated.

d) For methodD if array includes at least one number that is bigger than or equal to 4 then worst time complexity is $O(n)$.
If array does not includes a number that bigger than 4 then while loop goes infinite loop and worst time complexity cannot be determined.

3) "withoutLoop" method prints all of the elements of "myarray" by increasing local variable "i" one by one. So its time complexity is $O(n)$.

"WithLoop" method prints all of the elements of "myarray" by iterates through each element. So its time complexity is still $O(n)$.

But second method is more advantegous because it's more readable and if length of array changes first method cannot reach all elements. but second one can.

4) No we cannot solve this problem in constant time because we dont know where is our element in array,

```
function search(array, element)
    for(i=0 to array.length)
        if(array[i] = element)
            return 1
    return 0
```

This functions time complexity is $O(n)$ because of for loop.

5)

```
function findmin(A, B)
    min = 9999....
    n = A.length
    m = B.length
    for(i=0 to n)
        for(j=0 to m)
            if(a[i].bs < min)
                min = a[i].bs
    return min
```

⇒ This functions time complexity is $O(n.m)$ because of two for loops.