COSC222 Lab 3

Alexandre Rogov, Std-Id: 52162740

Q1) Show that:

a) $n^2 + 10$ *is* $O(n^2)$

$n^2 + 10 \leq cn^2$

$(n^2 + 10)/ n^2 \leq c$

Holds for c=11 and n_0=1

b) $(n+1)^3$ *is* $O(n^3)$

$(n+1)^3 \leq cn^3$

$(n+1)^3/ n^3 \leq c$

$n+1/n \leq c$

Holds for c= 2 and n_o=1

c) $2^{(n+1)}$ *is* $O(2^n)$

$2^{(n+1)} \leq c2^n$

$2^{(n+1)}/ 2^n \leq c$

$2^{(n+1)-n} \leq c$

$2 \leq c$

Holds for c=2 and n_o=1…inf

d) $3n^2 + 8nlog(n) + 3$ *is* $O(n^2)$

$3n^2 + 8nlog(n) + 3 \leq cn^2$

$(3n^2 + 8nlog(n) + 3)/n^2 \leq c$

Holds for c = 6 and n_o=1

Q2)

i) doubling n

ii) increase n by 1

a)*n^2*

i) Slower by a factor of 4

ii) Slower by a factor of 2 with an additive (1+2n)

b)*n^3*

i) Slower by a factor of 8

ii) Slower by a factor of 8 with an additive ($3n^2$+3n+1)

c)100*n^2*

i) Slower by a factor of 4

ii) Slower by a factor of 2 with an additive (n*100 - n^2) or multiply (100)

Q3)
$f1(n) = n^{3.5}$
f2(n) =sqrt(n)
f3(n) = n + 10
$f4(n) = 10n^2$
f5(n) = 2nlog(n)
$f6(n) = 2^n log(n)$

$f_3(n) < f_2(n) < f_5(n) < f_4(n) < f_1(n) < f_6(n)$

Q4)
a)$(n/2)*\log_2(n)$
Explanation: The first loop increments by i+2 which means it reaches the condition twice as fast or the condition is half as small (diving by 2). The second loop increments by multiple of two which means it has the log base of 2. If the condition of j was 2, we would only need one increment. Algebraically, can be written as $\log_2(2)=1$. Eg. If condition was 4 $(4/2)*\log_2(4) = 4$

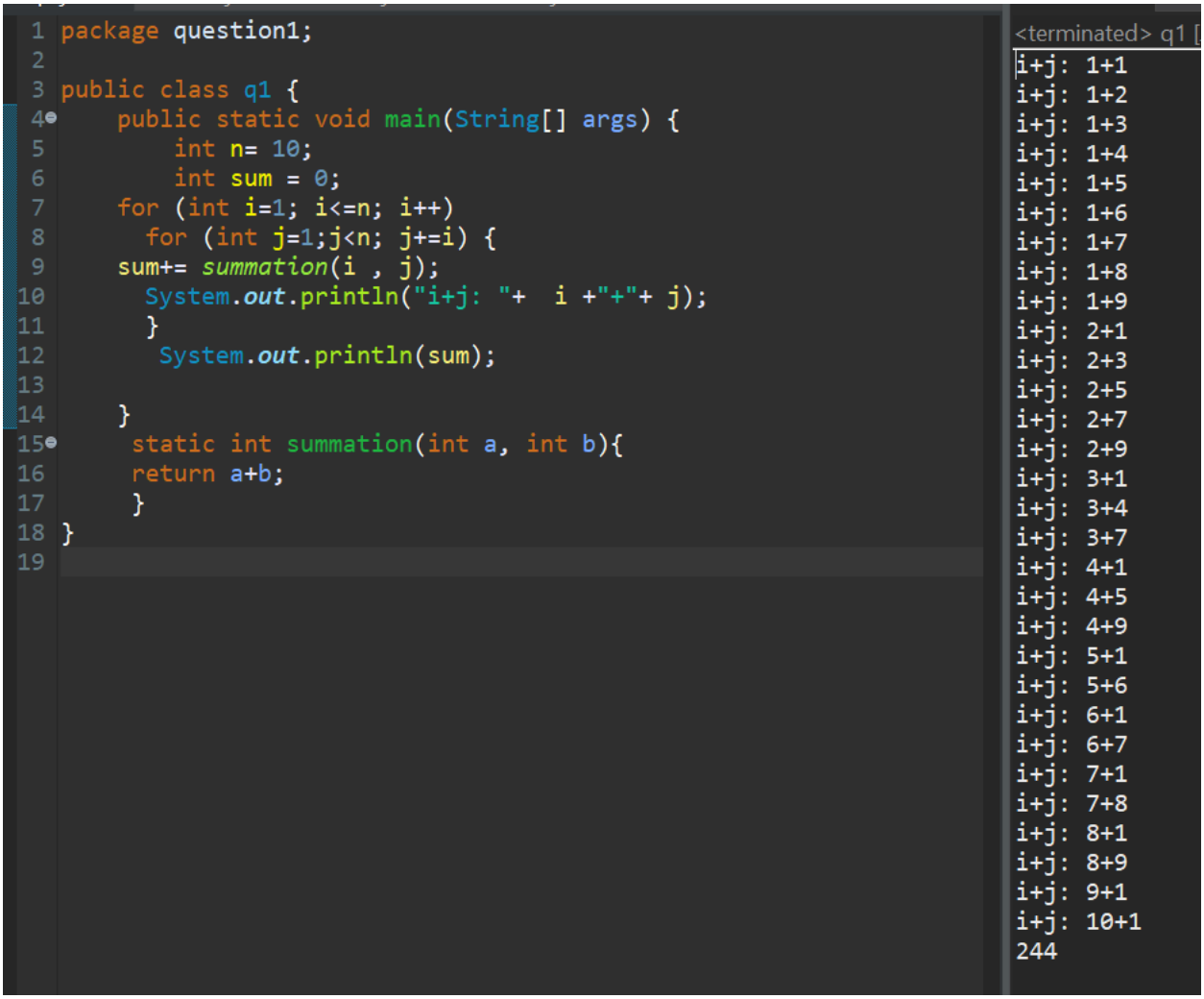
b)n
Explanation: Since there is a break every time in the second loop. We can ignore it. This leaves us with a standard for-loop which increments by 1. It would take n steps to reach n.

c)$n*\log_{10}(n)$
Explanation: The first loop is a plain n and is explained in b). The second loop increments by a multiple of 10 which means it has a log base of 10.e.g. If condition was 100, $100* \log_{10}(100) = 200$

d) n(log(n))
Explanation: The n on the outside corresponds to the outer loop. The inner loop increments by a changing value which converges to n. So for example, (n + n/2 + n/3 + …. n/n), pull the n out and you get log. Combine n and log(n). Then you get your answer. Voila, magic: n*log(n). This was pulled from stack Overflow. No example because it obviously doesn't work. But let's prove that incorrectness instead….

Eg. If n = 10, (10)*log(10) = 10

So, there should be 10 steps in the program. But Java will say otherwise:



```java
package question1;

public class q1 {
    public static void main(String[] args) {
        int n= 10;
        int sum = 0;
    for (int i=1; i<=n; i++)
      for (int j=1;j<n; j+=i) {
    sum+= summation(i , j);
      System.out.println("i+j: "+  i +"+"+ j);
      }
       System.out.println(sum);

    }
     static int summation(int a, int b){
     return a+b;
     }
}
```

```
<terminated> q1 [
i+j: 1+1
i+j: 1+2
i+j: 1+3
i+j: 1+4
i+j: 1+5
i+j: 1+6
i+j: 1+7
i+j: 1+8
i+j: 1+9
i+j: 2+1
i+j: 2+3
i+j: 2+5
i+j: 2+7
i+j: 2+9
i+j: 3+1
i+j: 3+4
i+j: 3+7
i+j: 4+1
i+j: 4+5
i+j: 4+9
i+j: 5+1
i+j: 5+6
i+j: 6+1
i+j: 6+7
i+j: 7+1
i+j: 7+8
i+j: 8+1
i+j: 8+9
i+j: 9+1
i+j: 10+1
244
```

Conclusion:

This is clearly more that “10 steps”. Therefore, the big Oh notation is wrong. Thus….

StackOverflow.com CAN be wrong and that some questions are better left unanswered. Q.E.D.