1) Horner's algorithm: O(n) 15
Naive algorithm: $O(n^2)$
In Horner's algorithm you save in getting the product x^i.

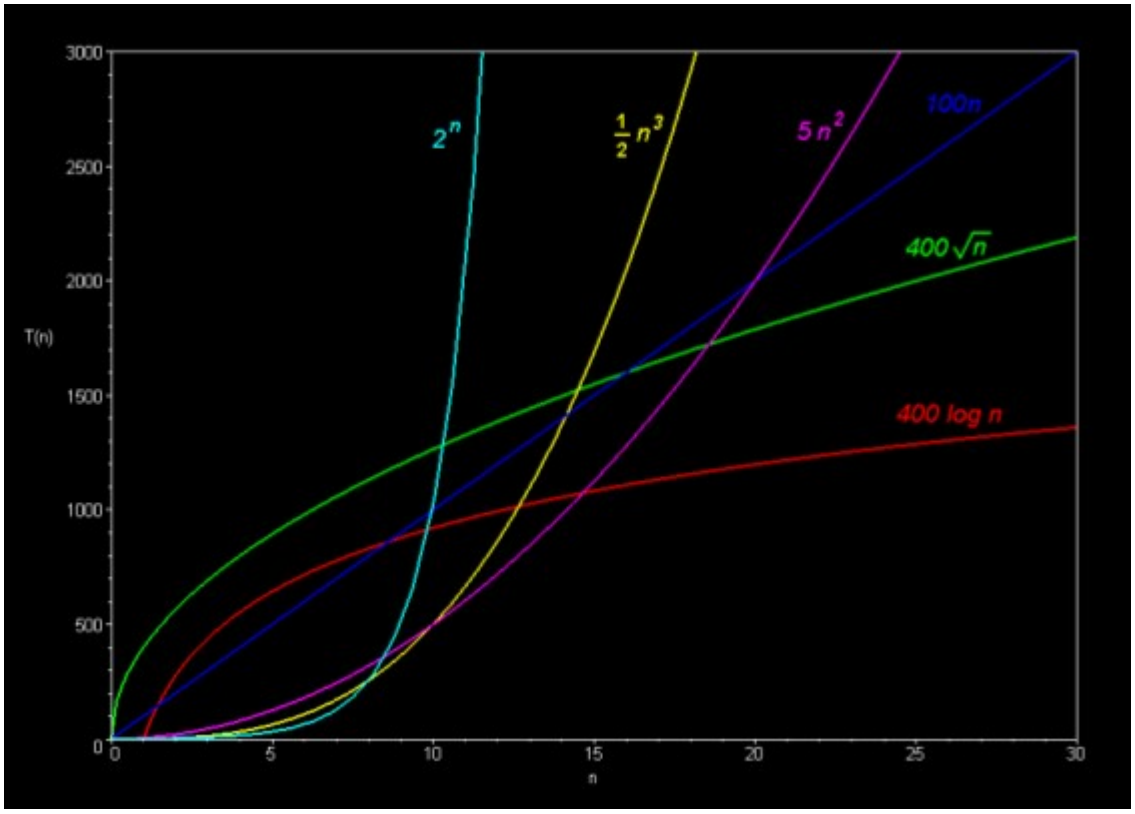
2) a) 15
main[
a[
b[
b finally
a caught Exc2
]a
b[
b finally
Program terminated due to Exception Exc2

b)
main[
a[
Program terminated due to Exception Exc1

3) Both add integers from 0 to n-1 in decreasing order to an empty list. 15
A: $O(n^2)$ - adds to the beginning and shifts other elements
B: O(n) – adds to the end and no shifting needs to be done

4) 15
```
public void forceAdd(int pos, E item) {
        try{
                add(pos, item);
        }
        catch(IndexOutOfBoundsException e){
                add( item);
        }
}
```

5) 20
While deciding which algorithm to opt for, you need to look at the size of your problem. As is seen in the graph below, lower order complexities dominate at small problem sizes but are overshadowed by the higher order terms as the size of the problem grows
a) A- $O(n^3)$
B- O(nlog(n))
C- $O(n^2)$
Rank- B,C,A
b) No. Sometimes constant terms are bigger.



6) a) O(m+n) 20

The inner loop add method is O(1) as it adds to the end. The If statement is O(n), the else is O(m) for remove and O(n) for the add.

b) $O(n^2)$

7) O(logn). Reduce the number of elementary operations by successively multiplying higher powers of *x* to get the result. 10