Lab 2

(1) Since we have a nested loop, which goes to n. We start to analyze from inside out. Hence the asymptotic running time is

$O(n^2)$ //

(2)

A, ALGORITHM merge (A, B)

INPUT: Arrays A and B of Sorted integers

OUTPUT: Array F of sorted elements of A and B Combined

```
LenA ← A.length
lenB ← B.length
n ← lenA + lenB
i ← 0
j ← 0

for x=0 to n-1 do
    if (i<lenA AND j<lenB) then
        IF(A[i]<B[j]) then
            F[x] ← A[i]
            Increment i
        ELSE
            F[x] ← B[j]
            Increment j
    ELSE IF (i<lenA) then
        F[x] ← A[i]
        Increment i
    ELSE IF (i<lenB) then
        F[x] ← B[j]
        Increment j

Return F
```

B. From the algorithm above, there are (15n+5) primitive operations and therefore the algorithm running time is $O(n)$

C)

```
package mergeTwo;
import java.util.Arrays;
public class Merge {
  public static void main(String [] args){
    int[] a={1, 4, 5, 8, 17};
    int [] b= {2, 4, 8, 11, 13, 21, 23, 25};
    System.out.println(Arrays.toString(merge(a,b)));
  }
  public static int [] merge(int[] a, int [] b){
    int n=a.length+b.length;
    int [] f = new int[n];
    for(int i=0,j=0,x=0;x<n;x++){
      if(i<a.length && j<b.length){
        if(a[i]<b[j])
        {
          f[x]=a[i];
          i++;
        }else
        {
          f[x]=b[j];
          j++;
        }
      }else if(i<a.length){
        f[x]=a[i];
        i++;
      }else if(j<b.length)
      {
        f[x]=b[j];
        j++;
      }
    }
    return f;
  }
```

Lab 2

(3)

(A) $\lim_{n\to\infty} \frac{1+4n^2}{n^2} = \lim_{n\to\infty}\left(4+\frac{1}{n^2}\right) = 4$ //

Hence, $1+4n^2$ is $O(n^2)$ //

(B) $\lim_{n\to\infty} \frac{n^2-2n}{n} = \lim_{n\to\infty}(n-2) = \infty$ //

Hence, n^2-2n is not $O(n)$ //

(C) $\lim_{n\to\infty} \frac{\log n}{n} = \frac{\infty}{\infty}$, we need L'Hopitals rule

Derivate the numerator & the denominator

$\lim_{n\to\infty} \frac{1}{n \ln 2} = \frac{1}{\infty} = 0$, Hence $\log(n)$ is $O(n)$ //

(D) $\lim_{n\to\infty} \frac{n}{n} = 1$, to be $O(n)$ the limit should be 0.

Hence n is not $O(n)$ //

Question 4

```
package powerset;
import java.util.*;
public class PowerSet {
  public static void main(String[] args){
    List <Integer> list = new ArrayList<>();
    list.addAll(Arrays.asList(1,4,5,6,7));
    PowerSet p=new PowerSet();
    System.out.println(p.powerSet(list) );
  }
  public List<Set<Integer>> powerSet(List<Integer> x){
    List<Set<Integer>> p = new ArrayList<>();
    Set<Integer> s=new TreeSet<>();
    p.add(s);
    Set<Integer> t;
    while (!x.isEmpty()) {
      Integer f=  x.remove(0);
      List<Set<Integer>> p1 = new ArrayList<>();
      for( Set i:p) {
        t= new TreeSet<>();
       t.addAll(i);
        t.add(f);
       p1.add(t);
      }
      p.addAll(p1);
    }
    return p;
  }
}
```