1. (2 point) Identify the complexity of the following code:

int f(int n) {   
  for (int i = n/3; i < n; i += 3) {   
    for (int j = 2\*n; j < 5\*n; j++) {   
      myfunc(); // assume this is O(1)  
    }  
  }   
}

// O(**\_\_\_\_\_\_\_\_\_\_\_\_\_**)

1. (3 points) Determine, for the typical algorithm that you use to perform calculations by hand, the run time to do the following:
2. Add two N-digit integers // O(**\_\_\_\_\_\_\_\_\_\_\_\_\_**)
3. Multiple two N-digit integers // O(**\_\_\_\_\_\_\_\_\_\_\_\_\_**)
4. Divid two N-digit integers // O(**\_\_\_\_\_\_\_\_\_\_\_\_\_**)
5. (3 points) What is the complexity of...
   1. Computing the average (mean) of a list of n numbers? O(\_\_\_\_\_\_\_\_\_\_\_\_\_)
   2. Finding if a number exists in a given list? O(\_\_\_\_\_\_\_\_\_) Ω(\_\_\_\_\_\_\_\_\_)
   3. Finding the nth element of a dynamic array (or grow array)? O(\_\_\_\_\_\_\_\_\_\_\_\_\_)
6. (4 points) What is the complexity of the following code:

void reverses(int[] array){

for (int i=0; i<array.length/2; i++){

int other=array.length-i-1;

int temp=array[i];

array[i]=array[other];

array[other]=temp;

}

}

// O(**\_\_\_\_\_\_\_\_\_\_\_\_\_**)

void printUnorderedPairs(int[] arrayA, int[] arrayB){

for (int i=0; i<arrayA.length; i++){

for (int j=0; j<arrayB.length; j++){

for (int k=0; k<10000; k++){

System.out.println(arrayA[i] + “,” + arrayB[j]);

}

}

}

}

// O(**\_\_\_\_\_\_\_\_\_\_\_\_\_**)

1. (8 points) Review the heavy-bottle problem discussed in class. Write pseudocode for a recursive solution. (Assumption: there is no limitation on how many times the scale can be used).