1. Consider the following implementation of binary search

```
// Assume A is sorted, so this is always true (invariant):
// A[start] <= x <= A[end-1]
public static int binarySearch(int[] A, int x) {
   int start = 0;
   int end = A.length;
   while (start < end) {
      int mid = (start + end) / 2;
      if (x == A[mid]) {
         return mid;
      } else if (x < A[mid]) {
           end = mid;
      } else {
           start = mid + 1;
      }
   }
   return -1;
}
```

(a) How many times does the line int mid = (start + end) / 2; execute when searching for value 3 in array [1,3,5,7,9,11]? Show your work.

(b) How many times does the line int mid = (start + end) / 2; execute when searching for value 0 in array [1,3,5,7,9,11]? Show your work.

2. Each of the following functions describes the number of constant time operations done by an algorithm as a function of the input size, N. For each, give the asymptotic runtime class for the corresponding function.
   (a) $15N + 2N^2 + 100$
   (b) $N \log N + 2N$
   (c) $2^N + N^2$
   (d) $101 + \log(N^2)$

3. For the following two problems, run the algorithms manually (without the aid of programs or tools).
   (a) If you sort the array 8, 3, 5, 4, 7 using insertion sort, how many times do you swap two elements? Show your work.

(b) If you sort the array 8, 3, 5, 4, 7 using selection sort, is the number of times you swap two elements greater, less than or equal to the number of swaps you came up with for insertion sort? Show your work.

8. Consider the following code. As a function of N, what is the total number of addition operations, assuming N is a power of 2.

```
public static int f4 (int n) {
    if (n <= 1) return 1;
    int i = 0;
    while (i < n) {
      i++;
    }
    return f4(n/2) + f4(n/2);
 }
```