CS 240: Growth Functions, Sigma Notation, and Algorithm Analysis

1. **Ordering Functions**

   Consider the following functions:

   - $A(n) = 1000$
   - $B(n) = \log_2(n)$
   - $C(n) = n \log_2(n)$
   - $D(n) = n^2$
   - $E(n) = 2^n$
   - $F(n) = n!$

   (a) **Without using a calculator** do your best to place the following values in increasing order from least to greatest:

   $$A(100{,}000{,}000) \quad B(8{,}000{,}000) \quad C(256) \quad D(2{,}000) \quad E(20) \quad F(15)$$

   Once you have made your best effort, check your answer using a calculator. If your calculator doesn't have a button for $\log_2$, recall the change of base rule: $\log_b a = \frac{\log_d a}{\log_d b}$

2. **Summations and Sigma Notation**

   (a) Explicitly write out each term in the following summations and calculate the resulting sums.

   $$\sum_{i=0}^{5} 10 = \qquad \sum_{i=1}^{5} i =$$

   $$\sum_{i=0}^{4} 2^i = \qquad \sum_{i=1}^{3} \sum_{j=1}^{2} ij =$$

   (b) Rewrite the following sum using sigma notation:

   $$3 + 4 + 5 + ... + n$$

   (c) Find the closed-form solution for your summation.

   (d) Check your solution by writing out the terms for $n = 6$ and comparing the result against your answer.

3. **Binary Search**

This is the binary search implementation provided by our textbook:

```
public static int binarySearch(int[] A, int K) {
  int low = 0;
  int high = A.length - 1;
  while(low <= high) {                      // Stop when low and high meet
    int mid = (low + high) / 2;             // Check middle of subarray
    if( A[mid] < K) low = mid + 1;          // In right half
    else if(A[mid] > K) high = mid - 1;     // In left half
    else return mid;                        // Found it
  }
  return A.length;                          // Search value not in A
}
```

For the questions below, assume that we are searching in the following 16-entry array:

3	8	11	26	28	33	50	57	68	71	79	84	85	90	91	98
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

(a) Trace the code above to determine the exact sequence of comparisons that will occur when we search for the following keys. Write down the values of `low`, `mid`, and `high` for each iteration.

- 71

- 27

(b) Find a key that will result in the maximum possible number of iterations. Trace the execution of binary search to determine how many array entries will be compared to this key. How does your answer compare to $\log_2(16)$?

4. **Algorithm Analysis**

For each of the following algorithms:

- Select an appropriate basic operation (double check your choice with your instructor).
- Develop a function that exactly maps the input size to the number of basic operations.
- If you have time... Put your code in a text editor and instrument it to check your answers.

(a)

```java
public static int maxTriple(int[] nums) {
  int biggest = -1;

  for (int i = 0; i < nums.length - 2; i++) {
    int sum = 0;
    for (int j = 0; j < 3; j++) {
      sum += nums[i + j];
    }
    if (sum > biggest) {
      biggest = sum;
    }
  }
  return biggest;
}
```

(b) Note the number of steps required for this example could depend on the values stored in the array, not just the length of the array. You should do a **worst-case** analysis: your function should describe the maximum number of steps possible for an input of the given size.

```java
public static int countDoubles(int[] nums) { int count = 0;

  for (int i = 0; i < nums.length; i++) {
    if (binarySearch(nums, nums[i] * 2) != nums.length) {
      count++;
    }
  }
  return count;
}
```

(c)

```java
public static void tweakNumbers(int[] nums) {

  // Add three everywhere
  for (int i = 0; i < nums.length; i++) {
    nums[i] += 3;
  }

  // Special entries get four more.
  for (int i = 1; i < nums.length; i *= 2) {
    nums[i] += 4;
  }
}
```