

6. Recall the worst-case time calculations for SelectionSort and InsertionSort:

[10 pts]

$$\text{SelectSort: } T_S(n) = \sum_{i=0}^{n-2} \left[c_1 + \left[\sum_{j=i+1}^{n-1} c_2 \right] + c_3 \right]$$

$$\text{InsertSort: } T_I(n) = \sum_{i=0}^{n-1} [c_1 + i(c_2 + c_3) + c_2 + c_4]$$

Suppose that:

- For a particular implementation of SelectionSort we have $c_1 = 0.6$, $c_2 = 0.6$, and $c_3 = 0.11$.
- For a particular implementation of InsertionSort we have $c_1 = 0.1$, $c_2 = 0.2$, and $c_3 = c_4 = 0.4$.

Even though both are $\Theta(n^2)$ it's possible for one to be faster for certain n . For which positive integer values of n will each method be faster and for any n might they take the same amount of time?

7. Consider the following algorithm:

```
\\ PRE: A is a global infinitely long strictly increasing list.
\\ PRE: TARGET is a target element which definitely exists in the list.
\\ PRE: n is a positive integer.
function binarysearch(A,TARGET,n)
    L = 0
    R = n
    while TARGET > A[R]
        R = 2 * R
    end
    while L <= R
        C = floor((L+R)/2)
        if A[C] == TARGET
            return C
        elif TARGET < A[C]
            R = C-1
        elif TARGET > A[C]
            L = C+1
        end
    end
    return FAIL
end
\\ POST:
```

- (a) Explain in your own words what this algorithm does and how it works. [10 pts]
- (b) Explain why the code will definitely achieve its goal. [5 pts]
- (c) If each assignment takes constant time c , explain why there is no upper limit to the amount of time this algorithm can take. [5 pts]