1. (30 pts) Professor Snape has n magical widgets that are supposedly both identical and capable of testing each other's correctness. Snape's test apparatus can hold two widgets at a time. When it is loaded, each widget tests the other and reports whether it is good or bad. A good widget always reports accurately whether the other widget is good or bad, but the answer of a bad widget cannot be trusted. Thus, the four possible outcomes of a test are as follows:

Widget A says	Widget B says	Conclusion
B is good	A is good	both are good, or both are bad
B is good	A is bad	at least one is bad
B is bad	A is good	at least one is bad
B is bad	A is bad	at least one is bad

   (a) Prove that if $n/2$ or more widgets are bad, Snape cannot necessarily determine which widgets are good using any strategy based on this kind of pairwise test. Assume a worst-case scenario in which the bad widgets are intelligent and conspire to fool Snape.

   (b) (Divide) Consider the problem of finding a single good widget from among the n widgets, assuming that more than $n/2$ of the widgets are good. Prove that $\lfloor n/2 \rfloor$ pairwise tests are sufficient to reduce the problem to one of nearly half the size.

   (c) (And Conquer!) Prove that the good widgets can be identified with $\Theta(n)$ pairwise tests, assuming that more than $n/2$ of the widgets are good. Give and solve the recurrence that describes the number of tests.

2. (25 pts) Professor Dumbledore needs your help. He gives you an array A consisting of n integers $A[1], A[2], \ldots, A[n]$ and asks you to output a two-dimensional $n \times n$ array B in which $B[i, j]$ (for $i < j$) contains the sum of array elements $A[i]$ through $A[j]$, i.e., the sum $A[i] + A[i + 1] + \cdots + A[j]$. (The value of array element $B[i, j]$ is left unspecified whenever $i \geq j$, so it doesn't matter what the output is for these values.)

   Dumbledore suggests the following simple algorithm to solve this problem:

```
dumbledoreSolve(A) {
   for i=1 to n
      for j = i+1 to n
         s = sum of array elements A[i] through A[j]
         B[i,j] = s
      end
   end
}
```

(a) For some function f that you should choose, give a bound of the form $O(f(n))$ on the running time of this algorithm on an input of size n (i.e., a bound on the number of operations performed by the algorithm).

(b) For this same function f, show that the running time of the algorithm on an input of size n is also $\Omega(f(n))$. (This shows an asymptotically tight bound of $\Theta(f(n))$ on the running time.)

(c) Although Dumbledore's algorithm is a natural way to solve the problem—after all, it just iterates through the relevant elements of B, filling in a value for each—it contains some highly unnecessary sources of inefficiency. Give an algorithm that solves this problem in time $O(f(n)/n)$ (asymptotically faster) and prove its correctness.

3. (20 pts) With a sly wink, Dumbledore says his real goal was actually to calculate and return the largest value in the matrix B, that is, the largest subarray sum in A. Butting in, Professor Hagrid claims to know a fast divide and conquer algorithm for this problem that takes only $O(n \log n)$ time (compared to applying a linear search to the B matrix, which would take $O(n^2)$ time).

   Hagrid says his algorithm works like this:

   - Divide the array A into left and right halves
   - Recursively find the largest subarray sum for the left half
   - Recursively find the largest subarray sum for the right half
   - Find largest subarray sum for a subarray that spans between the left and right halves
   - Return the largest of these three answers

   On the chalkboard, which appears out of nowhere in a gentle puff of smoke, Hagrid writes the following pseudocode for his algorithm:

```
hagridSolve(A) {
    if(A.length()==0) { return 0 }
    return hagHelp(A,1,A.length())
}

hagHelp(A, s, t) {
    if (s  > t) { return 0 }
    if (s == t) { return max(0, A[s]) }
```

```
    m = (s + t) / 2

    leftMax = sum = 0
    for (i = m, i > s, i--) {
            sum += A[i]
            if (sum >= leftMax) { leftMax = sum }
    }

    rightMax = sum = 0
    for (i = m, i <= t, i++) {
            sum += A[i]
            if (sum > rightMax) { rightMax = sum }
    }

    spanMax = leftMax + rightMax
    halfMax = max( hagHelp(s, m), hagHelp(m+1, t) )
    return max(spanMax, halfMax)
}
```

Hagrid claims that his algorithm is correct, but Dumbledore says "tut tut."

(i) Identify and fix the errors in Hagrid's code, (ii) prove that the corrected algorithm works, (iii) give the recurrence relation for its running time, and (iv) solve for its asymptotic behavior.

4. (10 pts) Suppose that we modify the `Partition` algorithm in QuickSort in such a way that on alternating levels of the recursion tree, `Partition` either chooses the best possible pivot or the worst possible pivot. Write down a recurrence relation for this version of QuickSort and give its asymptotic solution. Then, give a verbal explanation of how this `Partition` algorithm changes the running time of QuickSort.