1) (10pts) ANL (Algorithm Analysis)

Consider the recursive function `sum` shown below:

```
double sum(int* array, int low, int high){
    if (low == high)
        return array[low];
    int mid = (low+high)/2, left = 0, right = 0, i;
    for (i=low; i<=mid; i++) left += array[i];
    for (i=mid+1; i<=high; i++) right += array[i];
    if (left > right) return left + sum(array, low, mid);
    return right + sum(array, mid+1, high);
}
```

(a) (3 pts) Let T(n) represent the run time of the function call `sum(array, 0, n-1)`, where array is an integer array of size n. Write a recurrence relation that T(n) satisfies.

$$T(n) = T\left(\frac{n}{2}\right) + O(n), T(1) = 1$$

Grading: 2 pts T(n/2), 1 pt O(n).

(b) (7 pts) Using the iteration method, determine a closed-form solution (Big-Oh bound) for T(n). Assume T(1) = O(1).

$$T(n) = T\left(\frac{n}{2}\right) + cn$$
$$T(n) = T\left(\frac{n}{4}\right) + \frac{cn}{2} + cn$$
$$T(n) = T\left(\frac{n}{4}\right) + \frac{3cn}{2}$$
$$T(n) = T\left(\frac{n}{8}\right) + \frac{cn}{4} + \frac{3cn}{2}$$
$$T(n) = T\left(\frac{n}{8}\right) + \frac{7cn}{4}$$

After k steps, we have

$$T(n) = T\left(\frac{n}{2^k}\right) + \frac{(2^k - 1)cn}{2^{k-1}}$$

Plugging in $\frac{n}{2^k} = 1$, we have

$$T(n) = T(1) + \frac{(n-1)cn}{\frac{n}{2}}$$

$$T(n) = 1 + 2c(n-1)$$

$$T(n) = O(n)$$

Grading: 1 pt each iteration (3 pts total), 2 pts general form with k, 2 pts finishing problem. Note in place of *cn* students can write *O(n)* or even *n*, with no penalty.