

CIS121 - Spring 2008

Lab 2 – Monday/Tuesday, February 4/5

1. Derive a recurrence relation for the following piece of code and solve it:

```
public List<T> sort(List<T> listA, List<T> listB){
    List<T> result = new ArrayList<T>(listA.size() + listB.size());
    if(listA.size() > 1){
        int half = listA.size() / 2;
        listA = sort(listA.subList(0, half),listA.subList(half, listA.size()));
    }
    if(listB.size() > 1){
        int half = listB.size() / 2;
        listB = sort(listB.subList(0, half),listB.subList(half, listB.size()));
    }

    int indexA = 0, indexB = 0;
    while((indexA < listA.size()) && (indexB < listB.size())){
        if (listA.get(indexA).compareTo(listB.get(indexB)) <= 0){
            result.add(listA.get(indexA));
            indexA++;
        }
        else{
            result.add(listB.get(indexB));
            indexB++;
        }
    }

    if (indexA < listA.size())
        result.addAll(listA.subList(indexA, listA.size()));
    if (indexB < listB.size())
        result.addAll(listB.subList(indexB, listB.size()));
    }
    return result;
}
```

2. Solve the following recurrence relations:

(a) $T(1) = 0.5$ and $T(n) = 0.5n + T(n - 1)$.

(b) $T(1) = 1$ and $T(n) = 2T(n/2) + 1$.

You can assume that $n = 2^k$ for some integer k .

(c) $T(2) = 1$ and $T(n) = T(\sqrt{n}) + 1$.

You can assume that $n = 2^k$ for some integer k .