

CS106B Practice Midterm 2 Solutions

1. Recursion: Dorm Struggles

```
// Solution 1: use original function signature

Set<string> signRescue(string plan, int nFlakes) {
    string soFar;
    Set<string> result;
    helper(plan, nFlakes, soFar, result);
    return result;
}

// Slight variation uses an additional parameter saying the desired length
// and the base case checks against that. (In that case, do not need to pass
// the nFlakes parameter.)
void helper(string plan, int nFlakes, string soFar, Set<string>& result) {
    if (plan.length() == nFlakes) {
        if (isAllowed(soFar)) {
            result += soFar;
        }
    }
    for (int i = 0; i < plan.length(); i++) {
        helper(plan.substr(0, i) + plan.substr(i + 1), nFlakes, soFar + plan[i],
result);
    }
}

// Solution 2: use recursive helper function

Set<string> signRescue(string plan, int nFlakes) {
    return signHelper(plan, nFlakes, "");
}

Set<string> signHelper(string plan, int nFlakes, string soFar) {
    if (plan.size() == nFlakes) {
        if (isAllowed(soFar)) {
            return {soFar};
        }
    }

    Set<string> result;

    for (int i = 0; i < plan.size(); i++) {
        string rest = plan.substr(0, i) + plan.substr(i + 1);
```

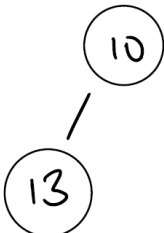
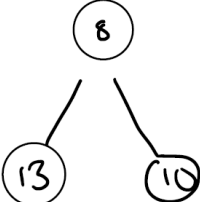
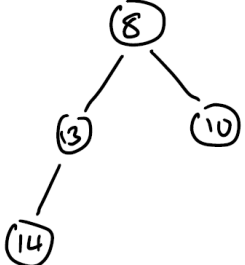
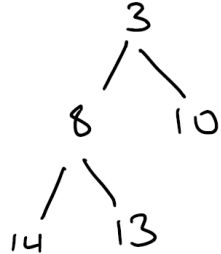
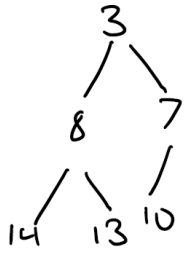
```

        result += signHelper(rest, nFlakes, soFar + plan[i], signs);
    }

    return result;
}

```

2. Priority Queue

Diagram after inserting 10:  <i>(this one is drawn for you as a format example)</i>	Diagram after inserting 13: 
Diagram after inserting 8: 	Diagram after inserting 14: 
Diagram after inserting 3: 	Diagram after inserting 7: 

3. BTS Dorm Trip

(a)

```
bool Tickets::isForSale(GridLocation seat) {  
    return _prices[seat] != NOT_FOR_SALE;  
}
```

(b)

C/C++

```
Set<GridLocation> Tickets::gatherGroup(GridLocation seat) {  
    Set<GridLocation> result;  
    Queue<GridLocation> toExplore = {seat};  
    while (!toExplore.isEmpty()) {  
        GridLocation next = toExplore.dequeue();  
        if (_prices[next] != NOT_FOR_SALE) {  
            _prices[next] = NOT_FOR_SALE;  
            result.add(next);  
            Set<GridLocation> adjacent = computeAdjacent(next);  
            for (GridLocation loc : adjacent) {  
                toExplore.enqueue(loc);  
            }  
        }  
    }  
    return result;  
}
```

(c)

C/C++

```
// make it work with original signature, no helper  
bool Tickets::findNSeatsInBudget(Vector<Set<GridLocation>>& groups,  
                                int N, int budget, Set<GridLocation>& result) {  
    if (totalCost(_prices, result) > budget || result.size() > N) {  
        return false;  
    }  
    if (result.size() == N) { // enough seats found  
        return true;  
    }  
}
```

```

    if (groups.size() == 0) { // out of options
        return false;
    }
    Set<GridLocation> cur = groups[groups.size() - 1];
    groups.remove(groups.size() - 1);
    // inclusion/exclusion pattern
    if (findNSeatsInBudget(groups, N, budget, result)) {
        return true;
    }
    result += cur;
    if (findNSeatsInBudget(groups, N, budget, result)) {
        return true;
    }
    result = result.difference(cur); // nuance: result -= cur doesn't
work
    groups.add(cur);

    return false;
}

```

4. Memory Diagram

```

C/C++
// (a)
KPop huntrix;
huntrix.size = 3;
huntrix.isDemon = false;
huntrix.enemy = new KPop;
huntrix.enemy->size = 5;
huntrix.enemy->isDemon = true;
huntrix.enemy->enemy = &huntrix;
string* lyrics = new string[3];
lyrics[0] = "Up Up Up";
lyrics[1] = "You're my soda pop";
lyrics[2] = "Gonna be golden";
KPop* saja;
saja = huntrix.enemy;
// (b)

```

```
delete [] lyrics;  
delete saja;
```

An alternative solution uses struct initialization:

```
C/C++  
//(a)  
KPop huntrix;  
huntrix.size = 3;  
huntrix.isDemon = false;  
huntrix.enemy = new KPop {5, true, &huntrix};  
string* lyrics = new string[3];  
lyrics[0] = "Up Up Up";  
lyrics[1] = "You're my soda pop";  
lyrics[2] = "Gonna be golden";  
KPop* saja;  
saja = huntrix.enemy;  
// (b) same as above
```