Homework 6: Q2

Name: Stephen Yang, syang29

*Don't forget to input your list of collaborators and sources on **AutoLab**.*
Please submit this file as a PDF.

1 Part (a): Algorithm Idea

```
placeInternetTowers(list houses){
        int max = 0;
        for(i=0 to i=houses size-1, i++){
                if(house i's distance from first house > max){
                        max = house i's distance
        }
        Return max/100

}
```

The return value is the number of towers. Place one tower at the first house, than the rest ahead of it each 100 miles apart.

2 Part (a): Runtime Analysis

O(n) for input
O(1) for max initialization
O(n) for loop in which if statement it O(1) assuming list access is O(1) (if list isn't arraylist we could have spent O(n) to put it into one outside the for loop so we could access it in O(1) here)
O(1) for return
O(n) for place one tower starting at first house and each 100 miles apart. Worst case scenario is that each house is 99 miles apart and thus each house (n amount) (basically) needs its Own tower (n/2 ish).
O(n) +O(1)+O(n)*O(1)+O(1)+O(n) = O(n)

3 Part (b): Algorithm Idea

```
putTowers(list houses){
        initialize arraylist<int> towerPlacement
        list.sort(houses)//sort houses by distance
        int needNextTower = 100;
        towerPlacement.add(0) //first house
        for(int i = 0 to i< houses.size, i++){
                if(houses.get(i).getDistance() >= needNextTower){
                        towerPlacement.add(houses.get(i).getDistance());
                        needNextTower = houses.get(i).getDistance()+100;
                }
        }
        Return towerPlacement
}
```

4 Part (b): Algorithm Details

Algorithm: Algorithm1

Algorithm idea should suffice. To summarize, order the input list by distance, keep knowledge of where the current tower loses service, once we get to a house outside of that service, that house has to have a tower next to it, in which we update the max mileage away from the first house that tower's service reaches. I return an arraylist stating the distance of the towers from the first house.

---

5 Part (b): Proof of Correctness Idea

To prove correct I have to 1)prove that there is no house that is over 100 miles ahead of the last tower and 2)that the algorithm gives the less amount of towers needed to provide full coverage.

I can prove by contradiction. Assume that there exists a house that is 100 miles away from the last tower. For that tower to be there it has to be in towerplacement. Theres only two scenarios where I put a tower in towerplacement.
a)putting the tower at distance 0

b)putting the tower next to a house that distance is greater than or equal to needNextTower

2) I can prove that this algorithm gives the least amount of towers need by proving by contradiction. Suppose there is an optimal solution that gives the least amount of towers needed. Assume that my algorithm returns more towers than that optimal solution.

That means that my algorithm puts a tower somewhere where it isn't needed. I will prove how this isn't possible

6 Part (b): Proof Details

1.a)In the case of the tower at distance 0, needNextTower is 100. needNextTower is only changed if the current house I am looking at is further than the needNextTower point (also a distance away from the first house). Once, I get to a house that is 100 or more miles away from the first house, I add a tower right next to that house. Anything before is less than 100 miles way from the first house since the list is ordered. That first house that is 100 or more miles way from the first house will thus have a tower right next to it so it isn't more than 100miles away from the last tower (it is 0 distance away) by the end of the iteration. The follows to the second case b.

b)In the case of putting a tower next to a house that distance is greater than or equal to needNextTower, to even get to that line, the if statement that line is in would have to be true, which means that the current house's distance from the first house is more than needNextTower point.

> The needNextTower point is always 100 miles in front of the last tower. The variable is only set twice. When it is initialized, because the first tower is at 0, 100 miles away is at 100 miles away (by definition). The other time it is set is in the if statement when the current house distance is greater or equal to needNextTower. When it is, I put a tower right there and set needNextTower to 100 more than that value.

Because for a tower to be put in (required for my contradiction), it had to be added to towerplacement. In this scenario where it is added in the, the if statement had to be true. If the if statement is true, than the algorithm will also go to the line where I update needNextTower. Whilst continuing to run the algorithm, once we get to a point where the if statement is true, (if it is false it is less than 100 miles away from previous tower), a tower is added right next to where that house is, which means it isn't 100 miles away from the last tower but actually 0 miles away. This is the contradiction.

2)Suppose there was an optimal algorithm that placed the least amount of towers needed for full coverage. Assume that my algorithm gives more towers than that amount. This means at some point there is a tower next to a house that doesn't need a tower there. For it to not need a tower there that means that there is a tower that is less than 100 miles away before it. But for that tower to be placed next to a house, an adding to the towerplacement list will have to be done, which is only done in two cases.

a)outside the for loop adding a tower to 0 away from the first house. This is needed because there is assumed to be no towers before it (or irrelevant) thus contradiction, tower is needed.

b)inside the for loop, inside the if statement, which means the if statement had to have been true. The if statement requires the current house's distance to be more or equal to the point that of needNextTower to be true. For that to be true, that means that house is more than or equal to 100 miles away from the previous tower based on the nature of needNextTower and how/when it is updated as proven in 1b. Since is it equal to or more than 100 miles away from the previous tower, it is not, by definition of more or equal to, less than 100 miles away. This is a contradiction.

7 Part (b): Runtime Analysis

O(n) to get input
O(1) to initialize towerPlacement
O(nlogn) to sort the list based on using an nlogn sort like merge sort, which is proven to be nlogn in the text
O(1) to initialize needNextTower and add the first tower into tower placement
O(n) for loop with an O(1) if statement (if access to house is O(1) in the case of an arraylist or an array, which if it isn't we could have took O(n) time outside the for loop to put the whole list inside an arraylist or array giving us O(1) access time anyway. It also has two statements to add to towerPlacement and updating needNextTower which is each O(1) statements. Return at its worst is O(n) since no house is more than 99 miles away. The rest of that proof/explanation is in ques 2 part a runtime analysis.

Added up O(n) +O(1)+ O(nlogn) + O(1) + O(n)*O(1) + O(n) = O(n)