HW4

705120273

Melody Chen

**Question 2:** This causes at least one compile error because our class of coordinates did not override the comparators for coordinates, so the compiler does not know how to compare the coordinates. In our insert function that takes only one parameter, it uses the > comparator, which we did not define in our coordinates class. It works for the other functions mentioned because for int, the > comparator is defined and for the other insert of coordinates, it did not use a comparator.

**Question 4B:** If we only had one parameter for this problem, we will not be able to come up with a solution because we will not have the ability to concatenate strings and pass them on to the next recursion. If string concatenation is not possible for this function, then it will be very difficult to make the parent of a node to be printed out for every children of the parent.

**Question 5A:**

const int N = *some value*;

bool isFriend[N][N];

...

int numMutualFriends[N][N];

for (int i = 0; i < N; i++) //O(N)

{

numMutualFriends[i][i] = -1; // the concept of mutual friend

// makes no sense in this case

for (int j = 0; j < N; j++) //O(N)

{

if (i == j)

continue;

numMutualFriends[i][j] = 0;

for (int k = 0; k < N; k++) //O(N)

{

if (k == i || k == j)

continue;

if (isFriend[i][k] && isFriend[k][j])

numMutualFriends[i][j]++;

}

}

}

The complexity for the algorithm above should be O(N3).

const int N = *some value*;

bool isFriend[N][N];

...

int numMutualFriends[N][N];

for (int i = 0; i < N; i++) O(N2)

{

numMutualFriends[i][i] = -1; // the concept of mutual friend

// makes no sense in this case

for (int j = 0; j < **i**; j++) O(iN) **// loop limit is now i, not N**

{

numMutualFriends[i][j] = 0;

for (int k = 0; k < N; k++) O(N)

{

if (k == i || k == j)

continue;

if (isFriend[i][k] && isFriend[k][j])

numMutualFriends[i][j]++;

}

**numMutualFriends[j][i] = numMutualFriends[i][j];**

}

}

The complexity