Homework 4

cs 32

Michelle navarrete Vega

# 2.

When attempting to insert a Coord into the Map, the Map checks if that coord already exists in there by using the == operator and checks ordering using the < operator. For the Coord class, however, neither operator has been defined. As a result, using them results in an error.

# 3E.

When the loop is first established, the pointer ‘p’ is given specific constraints and thus ends the loop when k = 5. Thus, when we add more items to our vector, they cannot be accessed as they are out of the range of the initial constraints given while in the loop.

# 4b.

If only one parameter had been given, recursive calls would be unable to determine their location; for example, a recursive call to Peach would not be aware that the path to itself was Actor=>, so it would be unable to print the full path to itself. Because recursion allows the function to know its path to it and branch off to each subclass, the result should be displayed recursively.

# 5a.

The time complexity of this algorithm would be O(N^3) because the 3 nested for loops each loop for N number of times.

int hasContacted[N][N];

int numIntermediaries[N][N];

for (int i = 0; i < N; i++) <---------------------- O(N^3)

{

numIntermediaries[i][i] = -1; <---------------------- O(1)

for (int j = 0; j < N; j++) <----------------------------O(N^2)

{

if (i == j) <------------------------------------------------O(1)

continue;

nunumIntermediaries[i][j] = 0; <---------------------- O(1)

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

{

if (k == i || k == j) <------------------------------- O(1)

continue;

if (hasContacted[i][k] && hasContacted[k][j]) <--------------O(1)

numIntermediaries[i][j]++;<------------------- O(1)

}

}

# 5b.

The time complexity of this algorithm would still be 0(N^3). The time complexity of the 5a and 5b both are 0(N^3) due to the 3 nested for-loops, but this algorithm would have a lower constant of proportionality because of the 2nd for loop going till ‘i’ instead of ‘N’ and because of the line “numMutualFriends[j][i] = numMutualFriends[i][j].” Because the constants of proportionalities and lower order terms are ignored, the two algorithms have the same proportionality constant.

bool hasContacted[N][N];

int numIntermediaries[N][N];

for (int i = 0; i < N; i++) <----------------------------O(N^3)

{ <----------------------------O(N^2)

numIntermediaries[i][i] = -1; <---------------------------- O(1)

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

{ <---------------------------------------------------- O(N)

numIntermediaries[i][j] = 0; <---------------------------- O(1)

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

{

if (k == i || k == j) <---------------------------- O(1)

continue;

if (hasContacted[i][k] && hasContacted[k][j]) <---------------------------- O(1)

numIntermediaries[i][j]++; <---------------------------- O(1)

}

**numIntermediaries[j][i] = numIntermediaries[i][j];** <------ O(1)

}

}

# 6a.

The time

# 6B.

The time

# 7.