BusGraph

```
public class BusGraph {
   LinkedList<CityVertex> cities;

   public BusGraph() {
       this.cities = new LinkedList<CityVertex>();
   }

   public void addCity(CityVertex cv) {
       this.cities.add(cv);
   }

   public void addRoute(CityVertex fromCity, CityVertex toCity) {
       if (this.cities.contains(fromCity) &&
               this.cities.contains(toCity)) {
            fromCity.addEdge(toCity);
       }
       // throw a CityNotFound exception otherwise
   }

   public boolean canReach(CityVertex source, CityVertex dest) {
       return source.canReach(dest);
   }

}
```

CityVertex

```
public class CityVertex {
   LinkedList<CityVertex> toCities;
   String name;

   public CityVertex(String nm) {
       this.name = nm;
       this.toCities = new LinkedList<CityVertex>();
   }

   public void addEdge(CityVertex toVertex) {
       this.toCities.add(toVertex);
   }









   public boolean canReach(CityVertex dest) {
       if (this.equals(dest)) {
           return true;
       }


       for (CityVertex neighbor : this.toCities) {


           if (neighbor.canReach(dest)) {
               return true;
           }
       }
       return false;
   }




   public String toString() {
       String retstring = "City " + this.name + " goes to { ";
       for (CityVertex toCity : this.toCities) {
           retstring += toCity.name + " ";
       }
       retstring += "}";
       return retstring;
   }
}
```

TestCityGraph

```
public class TestCityGraph {
    BusGraph neBuses;
    CityVertex man, bos, pvd, wos, har;

    @Before
    public void setup() {
        neBuses = new BusGraph();

        man = new CityVertex("Manchester");
        neBuses.addCity(man);
        bos = new CityVertex("Boston");
        neBuses.addCity(bos);
        pvd = new CityVertex("Providence");
        neBuses.addCity(pvd);
        wos = new CityVertex("Worcester");
        neBuses.addCity(wos);
        har = new CityVertex("Hartford");
        neBuses.addCity(har);

        neBuses.addRoute(man, bos);
        neBuses.addRoute(bos, pvd);
        neBuses.addRoute(bos, wos);
        neBuses.addRoute(pvd, bos);
        neBuses.addRoute(wos, har);
    }

    @Test
    public void testCanReach() {
        System.out.println(man);
        System.out.println(bos);









    }

}
```

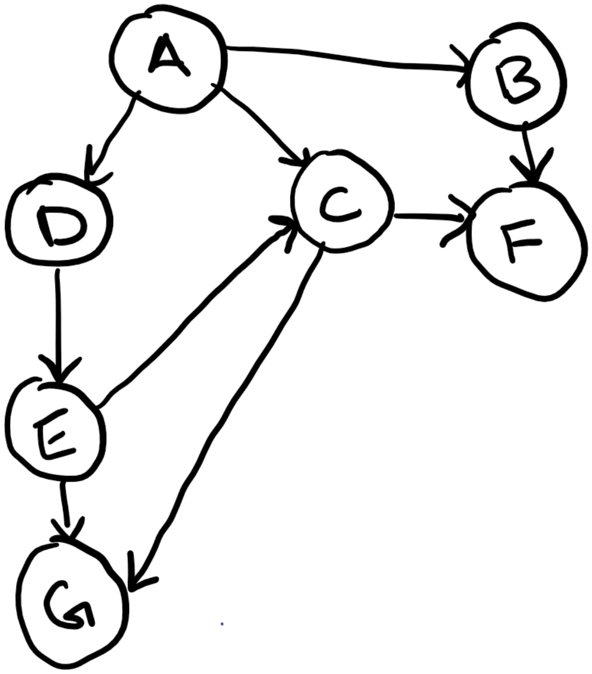
Method calls for canReach:

bos.canReach(har)

Method calls for canReachHelper:

bos.canReachHelper(har, {})

Keeping track using a LinkedList:



DFS visited: DFS list:	BFS visited: BFS list: