CO2226 : Algorithm Design and Analysis
Practical 2
Adjacency Matrix
(Adapted from CW1415/1516)

1. Below is a program that was written by the Examiner. Play with the program and alter the graph and check that you understand how the program works.

```
import java.util.HashSet;
import java.util.ArrayList;
public class graph
{
  double [] [] adj;
  graph (double [] [] a)
  {
   adj= new double [a.length][a.length];
   for (int i=0;i<a.length;i++)
     for (int j=0;j<a.length;j++)
        adj[i][j]=a[i][j];
  }

  public HashSet <Integer> neighbours(int v)
  {
   HashSet <Integer> h = new HashSet <Integer> () ;
   for (int i=0;i<adj.length;i++) if (adj[v][i]!=0) h.add(i);
   return h;
  }

  public HashSet <Integer> vertices()
  {
   HashSet <Integer> h = new HashSet <Integer>();
   for (int i=0;i<adj.length;i++) h.add(i);
   return h;
  }

  ArrayList <Integer> addToEnd (int i, ArrayList <Integer> path)
  // returns a new path with i at the end of path
  { ArrayList <Integer> k;
   k = (ArrayList<Integer>) path.clone() ;
   k.add(i);
   return k;
  }
```

```
public HashSet  <ArrayList <Integer>> shortestPaths1 ( HashSet <ArrayList <Integer>> sofar,
                                                       HashSet <Integer> visited, int end)
{
 HashSet <ArrayList <Integer>> more = new HashSet <ArrayList <Integer>>();
 HashSet <ArrayList <Integer>> result = new HashSet <ArrayList <Integer>>();
 HashSet <Integer> newVisited = (HashSet <Integer>) visited.clone();
 boolean done = false;
 boolean carryon = false;
 for (ArrayList <Integer> p: sofar)
 {
  for (Integer z: neighbours(p.get(p.size()-1)))
  {
    if (!visited.contains(z))
    {
     carryon=true;
     newVisited.add(z);
     if (z==end) {done=true; result.add(addToEnd(z,p));}
     else more.add(addToEnd(z,p));
    }
  }
 }
 if (done) return result;
 else
  if (carryon) return shortestPaths1(more,newVisited,end);
  else return new HashSet <ArrayList <Integer>>();
}

public HashSet <ArrayList <Integer>> shortestPaths(int first, int end)
{
  HashSet <ArrayList <Integer>> sofar = new HashSet <ArrayList <Integer>>();
  HashSet <Integer> visited = new HashSet <Integer>();
  ArrayList <Integer> starting = new ArrayList <Integer>();
  starting.add(first);
  sofar.add(starting);
  if (first==end) return sofar;
  visited.add(first);
  return shortestPaths1(sofar,visited,end);
}
```

```
public static void main(String [] args)
   {
    double [ ] [] a = {
                    {0.0, 1.0, 1.0, 0.0},
                    {0.0, 0.0, 1.0, 1.0},
                    {0.0, 1.0, 0.0, 1.0},
                    {0.0, 1.0, 1.0, 0.0}
                   };
     graph g = new graph(a);
     for (int i=0;i<a.length;i++)
     {  for (int j=0;j<a.length;j++)
          if (i!=j) System.out.println(i + " to " + j +": "+g.shortestPaths(i,j));
     }
   }
}
```

Expected Output :

```
0 to 1: [[0, 1]]
0 to 2: [[0, 2]]
0 to 3: [[0, 2, 3], [0, 1, 3]]
1 to 0: []
1 to 2: [[1, 2]]
1 to 3: [[1, 3]]
2 to 0: []
2 to 1: [[2, 1]]
2 to 3: [[2, 3]]
3 to 0: []
3 to 1: [[3, 1]]
3 to 2: [[3, 2]]
```

2. The London Underground System

Study the following files of data about the London Underground System:

a. Some information about the stations: stations

"id","latitude","longitude","name", "display_name","zone","total_lines","rail“

Only the first four fields of every line will be used in this assignment. The last four are not needed.

b. Some information about which stations are adjacent: edges

"station1","station2","line“

Only the first two fields of every line will be used in this assignment.

For example the first line 11,163,1 means that station 11 (Baker Street) and 163 (Marylebone) are adjacent. (Ignore the 1!)

c. Play with the following program:

```
import java.io.*;
import java.util.Scanner;
import java.util.*;
class underground
{  static int N= 500;
   static double [][] edges = new double[N][N];
   static TreeMap <Integer,String> stationNames = new TreeMap <Integer,String>();
   static ArrayList<String> convert (ArrayList<Integer> m)
  {      ArrayList<String> z= new ArrayList<String>();
         for (Integer i:m) z.add(stationNames.get(i));
         return z;
  }
  static HashSet<ArrayList<String>> convert (HashSet<ArrayList<Integer>> paths)
 {     HashSet <ArrayList <String>> k= new HashSet <ArrayList <String>>();
       for (ArrayList <Integer> p:paths) k.add(convert(p));
            return k;
  }
public static void main(String[] args) throws Exception
  {
    for(int i=0;i<N;i++)for(int j=0;j<N;j++) edges[i][j]=0.0;
    Scanner s = new Scanner(new FileReader("edges"));
    String z =s.nextLine();
    while (s.hasNext())
    {
      z =s.nextLine();
      String[] results = z.split(",");
      edges[Integer.parseInt(results[0])][Integer.parseInt(results[1])]=1.0;
      edges[Integer.parseInt(results[1])][Integer.parseInt(results[0])]=1.0;
    }
    s = new Scanner(new FileReader("stations"));
    z =s.nextLine();
    while (s.hasNext())
    {
     z =s.nextLine();
     String[] results = z.split(",");
     stationNames.put(Integer.parseInt(results[0]),results[3]);
    }
    graph G= new graph (edges);
    int st = Integer.parseInt(args[0]);
    int fin = Integer.parseInt(args[1]);
    System.out.println("Shortest path from " + stationNames.get(st) + " to " +
        stationNames.get(fin) + " is " + convert(G.shortestPaths(st,fin)));
  }
}
```

Expected Output :

```
Shortest path from South Ruislip to Upminster Bridge is [[South Ruislip, Northolt,
Greenford, Perivale, Hanger Lane, North Acton, East Acton, White City, Shepherd's
Bush (C), Holland Park, Notting Hill Gate, Queensway, Lancaster Gate, Marble Arch,
Bond Street, Green Park, Westminster, Waterloo, Bank, Liverpool Street, Bethnal Green,
Mile End, Stratford, West Ham, Plaistow, Upton Park, East Ham, Barking, Upney,
Becontree, Dagenham Heathway, Dagenham East, Elm Park, Hornchurch, Upminster
Bridge]]
```

d. Write a Java program, using the shortestPaths method above, to obtain answers for the following :

1. How many shortest paths are there between South Ruislip (239) and Upminster (268)?

   Ans : 1

2. What is the length (number of stations including both ends) of the shortest path between Holborn and Holloway Road? Hint: You may wish to use the following method.

```
static ArrayList<Integer> firstElement (HashSet <ArrayList <Integer>> s)
{   return (ArrayList<Integer>)s.toArray()[0]; }
```

   Ans : 6

3. Which pair of stations have the highest number of shortest paths between them?

   Ans : [29] [242]

4. How many shortest paths do they have?

   Ans : 24

5. Which set of stations are furthest away from station 13 in terms of number of stops? (Just print out a set of numbers corresponding to the stations.)

   Ans : [117, 118]