SOLUTION NOTES

Foundations of Programming 2001 Paper 11 Question 2 (FHK)

Each of the vertices 0 to 7 in the graph leads to 1 or 2 other vertices. If vertex n leads to two vertices, then first(n) is the lower-numbered of the two and second(n) is the higher numbered of the two.

If vertex n leads to one vertex, then first(n) is the number of that vertex and second(n) is zero.

This is a perfectly adequate data structure for the given graph but for several reasons it is hardly a sound basis for a general-purpose description of a directed graph...

First, since there are only two arrays, a vertex may lead to at most two other vertices.

Secondly, it is assumed that no vertex leads to vertex 0 (otherwise 0 could not be used as it is in array second).

Finally, it is fortuitous that the terminal vertex is vertex 8 and there is thus no need for elements indexed by 8.

[5 marks]

A complete test program is given below. Candidates are required to write only the body of tryit(). The operation is reasonably straightforward. Starting at vertex 0 (via the call of tryit(0) in the print statement) each vertex is visited in turn. There are two possibilities...

If vertex n has been visited before (or is vertex 8), the value of state(n) [the ‘score’ for vertex n] indicates how many routes there are from n to 8 and this value is immediately returned.

If vertex n has not been visited before, the number of routes from n to 8 is calculated by visiting the one or two vertices to which n leads and noting its score or their scores. Visiting is recursive in an obvious way.

```
public class Routes
 { private static final int[] first = {1,3,4,6,6,7,8,8};
   private static final int[] second = {2,4,5,0,7,0,0,0};
```

```
  private static int[] state = {-1,-1,-1,-1,-1,-1,-1,-1,+1};

  public static void main(String[] args)
   { System.out.println("There are " + tryit(0) + " routes");
   }

  private static int tryit(int vertex)
   { int score = state[vertex];
     if (score<0)
      { score = 0;
        score += tryit(first[vertex]);
        if (second[vertex]>0)
           score += tryit(second[vertex]);
        state[vertex] = score;
      }
     return(score);
   }
}
```

[15 marks]