Graph – 19 points; Suggested work & submission time less than 10min

Submission instructions:
Please submit a single file (use any file name) on TEACH under FinalExamProblem4 before 11am. You may write answers by hand and submit a scanned copy. There is no need to copy the questions; just **list the question numbers next to your answers.**

Answer the following two multi-choice questions. The correct answers bring 7 and 12 points, respectively. An incorrect answer will be penalized by **-5 points** to discourage random guessing. The minimum possible score is 0.

We are given a binary adjacency matrix `M` of a graph with `n` nodes, where `M[u][v] = 1` if there is an edge from node `u` to node `v`; and `M[u][v] = 0` otherwise.

From the four pieces of code, **select all correct** codes for computing the following tasks.

- **Task 1 – 7 points:** Compute the reachability matrix `X`, where `X[u][v] = 1` if node `v` is reachable from node `u`; and `X[u][v] = 0` otherwise.

[✓ CODE 1:]

```
for(u = 0; u < n; u++)
 for(v = 0; v < n; v++)
  for(a = 0; a < n; a++)
   X[u][v] = X[u][v] + X[u][a] * M[a][v];
```

[✓ CODE 2:]

```
for(b = 0; b < n; b++)
 for(u = 0; u < n; u++)
  for(v = 0; v < n; v++)
   X[u][v] = X[u][v] + X[u][b] * M[b][v];
```

CODE 3:

```
for(u = 0; u < n; u++)
 for(c = 0; c < n; c++)
  for(v = 0; v < n; v++)
   X[u][v] = X[u][v] + X[u][c] * M[c][v];
```

CODE 4:

```
for(d = 0; d < n; d++)
 for(u = 0; u < n; u++)
  for(v = 0; v < n; v++)
   X[u][v] = X[u][v] + X[u][v] * M[d][v];
```

- **Task 2 – 12 points:** For a given start node u, and for every node v in the graph, compute the minimum cost c[v] of a path from u to v.

CODE 1:
```
#define INFINITY 10000
for(v = 0; v < n; v++){
  visited[v] = 1;
  c[v] = M[u][v];
}
min = 0;
while (min < INFINITY){
  min = INFINITY;
  for(v = 0; v < n; v++)
    if(min > c[v]){
      min = c[v];
      minIdx = v;
    }
  visited[minIdx] = 0;
  /*update neighbors of minIdx*/
  for(v = 0; v < n; v++)
    if (visited[v] && M[minIdx][v])
      if(c[v] > c[minIdx] + M[minIdx][v])
        c[v] = c[minIdx] + M[minIdx][v];
}
```

CODE 2:
```
#define INFINITY 10000
for(v = 0; v < n; v++){
  visited[v] = 1;
  c[v] = M[u][v];
}
min = 0;
while (min < INFINITY){
  min = INFINITY;
  for(v = 0; v < n; v++)
    if(visited[v] && min > c[v]){
      min = c[v];
      minIdx = v;
    }
  visited[minIdx] = 0;
  /*update neighbors of minIdx*/
  for(v = 0; v < n; v++)
    if (M[minIdx][v])
      if(c[v] > c[minIdx] + M[minIdx][v])
        c[v] = c[minIdx] + M[minIdx][v];
}
```

CODE 3:
```
#define INFINITY 10000
for(v = 0; v < n; v++){
  visited[v] = 0;
  c[v] = M[u][v];
}
min = 0;
while (min < INFINITY){
  min = INFINITY;
  for(v = 0; v < n; v++)
    if(!visited[v] && min > c[v]){
      min = c[v];
      minIdx = v;
    }
  /*update neighbors of minIdx*/
  for(v = 0; v < n; v++)
    if (!visited[v] && M[minIdx][v])
      if(c[v] > c[minIdx] + M[minIdx][v]){
        c[v] = c[minIdx] + M[minIdx][v];
        visited[v] = 1;
      }
}
```

CODE 4:
```
#define INFINITY 10000
for(v = 0; v < n; v++){
  visited[v] = 0;
  c[v] = M[u][v];
}
min = 0;
while (min < INFINITY){
  min = INFINITY;
  for(v = 0; v < n; v++)
    if(!visited[v] && min > c[v]){
      min = c[v];
      minIdx = v;
    }
  visited[minIdx] = 1;
  /*update neighbors of minIdx*/
  for(v = 0; v < n; v++)
    if (M[minIdx][v])
      if(c[v] > c[minIdx] + M[minIdx][v]){
        c[v] = c[minIdx] + M[minIdx][v];
        visited[v] = 1;
      }
}
```