Christiaan Cronje

COSC3319.01

8 a.m.

March 2, 2017

Lab 1 Grading Option “A”

outputC.txt

A B C D J K L

-------------------------------------------

A | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

-------------------------------------------

B | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

-------------------------------------------

C | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

-------------------------------------------

D | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

-------------------------------------------

J | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

-------------------------------------------

K | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

-------------------------------------------

L | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

-------------------------------------------

outputB.txt

A B C D J K L

-------------------------------------------

A | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

-------------------------------------------

B | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

-------------------------------------------

C | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

-------------------------------------------

D | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

-------------------------------------------

J | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

-------------------------------------------

K | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

-------------------------------------------

L | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

-------------------------------------------

outputA1.txt

Ada Bob Joe Ken Sam Sue Tim Tom Jim

-------------------------------------------------------

Ada | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

-------------------------------------------------------

Bob | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

-------------------------------------------------------

Joe | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |

-------------------------------------------------------

Ken | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |

-------------------------------------------------------

Sam | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

-------------------------------------------------------

Sue | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

-------------------------------------------------------

Tim | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

-------------------------------------------------------

Tom | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |

-------------------------------------------------------

Jim | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

-------------------------------------------------------

outputA2.txt

1 2 3 4 5 6 7

-------------------------------------------

1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

-------------------------------------------

2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

-------------------------------------------

3 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

-------------------------------------------

4 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

-------------------------------------------

5 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

-------------------------------------------

6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

-------------------------------------------

7 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

-------------------------------------------

Option A Code

/\*----------Matrix.cs--------\*/

using System.IO;

namespace Warshall

{

class Matrix

{

//Create Matrix of generic type

public static T[,] CreateMatrix<T>(int matrixSize, T falseValue)

{

T[,] tempMatrix = new T[matrixSize, matrixSize];

//default all links to 0

for (int i = 0; i < matrixSize; i++)

{

for (int j = 0; j < matrixSize; j++)

{

tempMatrix[i, j] = falseValue;

}

}

return tempMatrix;

}

public static string[] CreateMatrixNames(int size)

{

return new string[size];

}

//Use Warshall's Algorithm to mark nodes that can communicate with each other

public static void GetLinkedNodes<T>(ref T[,] matrix, T trueValue, T falseValue)

{

int matrixLength = matrix.GetLength(0);

for (int i = 0; i < matrixLength; i++)

{

for (int j = 0; j < matrixLength; j++)

{

if (object.Equals(matrix[j, i], trueValue))

{

for (int k = 0; k < matrixLength; k++)

{

matrix[j, k] = or(matrix[j, k], matrix[i, k], trueValue, falseValue);

}

}

}

}

}

//No operator overloading in C#

//Function to handle the "or" operation

static T or<T>(T i, T j, T trueValue, T falseValue)

{

T result = falseValue;

if (object.Equals(i, trueValue) || object.Equals(j, trueValue))

result = trueValue;

return result;

}

//read from file

public static string[] ReadInputFromFile(string fileName)

{

string[] fileContent;

//open, read, and close file

fileContent = File.ReadAllLines(fileName);

return fileContent;

}

//Create matrix in single string format

public static string ToString<T>(T[,] matrix, string[] names)

{

string outstring = "";

int n = matrix.GetLength(0);

outstring += "\n\t";

for (int j = 0; j < n; j++)

{

outstring += names[j] + "\t";

}

outstring += "\n";

for (int j = 0; j < 6 \* (n) + 1; j++)

{

outstring += "-";

}

for (int i = 0; i < n; i++)

{

outstring += "\n" + names[i] + " |\t";

for (int j = 0; j < n; j++)

{

outstring += matrix[i, j] + " |\t";

}

outstring += "\n";

for (int j = 0; j < 6 \* (n) + 1; j++)

{

outstring += "-";

}

}

outstring += "\n\n";

return outstring;

}

//Output text to specified file

public static void WriteToFile(string fileName, string output)

{

output = fileName + "\n\n" + output;

File.WriteAllText(fileName, output);

}

}

}

/\*-------LinkedNodes.cs---------\*/

using System;

using System.IO;

namespace Warshall

{

class LinkedNodes

{

public static void Main(string[] args)

{

string resp = "n";

do

{

string fileInName = "", fileOutName = "";

int matrixSize = 0;

//get number of nodes to determine the matrix size

Console.Write("Enter the number of nodes: ");

matrixSize = Convert.ToInt32(Console.ReadLine());

//create matrix

int[,] nodeMatrix = Matrix.CreateMatrix<int>(matrixSize, 0);

string[] nodeNames = Matrix.CreateMatrixNames(matrixSize);

//name the nodes

Console.Write("Would you like to name the nodes? [y/n]: ");

resp = Console.ReadLine();

if (resp.ToUpper().Equals("Y"))

{

for (int i = 0; i < matrixSize; i++)

{

Console.Write("Enter name for node " + (i + 1) + ": ");

nodeNames[i] = Console.ReadLine();

}

}

else {

//if not naming nodes, assign them numbers

for (int i = 0; i < matrixSize; i++)

{

nodeNames[i] = "" + (i + 1);

}

}

do

{

Console.Write("Enter input file name: ");

fileInName = Console.ReadLine();

} while (!File.Exists(fileInName));

string[] input = Matrix.ReadInputFromFile(fileInName);

//blanks are i \* size+1

//4x4 = 0, 5, 10, 15

//get input for relationship status of each node (except for selves i.e. node[2,2])

int readOffset = 0;

for (int i = 0; i < matrixSize; i++)

{

for (int j = 0; j < matrixSize; j++)

{

if (i == j)

{

//node i=j is always blank so no confirmation in needed

readOffset--;

continue;

}

//can node i talk to node j?

resp = "n";

int index = (i \* matrixSize) + j + readOffset;

Console.WriteLine(index);

if (resp.ToUpper().Equals("Y"))

{

nodeMatrix[i, j] = 1;

}

}

}

//print provided matrix to console for viewing

Console.WriteLine(Matrix.ToString(nodeMatrix, nodeNames));

//Run Warshall's Algorithm to find linked nodes

Matrix.GetLinkedNodes<int>(ref nodeMatrix, 1, 0);

//Request output file

do

{

Console.Write("Enter output file name: ");

fileOutName = Console.ReadLine();

resp = "y";

if (File.Exists(fileOutName))

{

Console.WriteLine("This file already exists.");

Console.Write("Do you wish to overwrite it? [y/n]: ");

resp = Console.ReadLine();

}

} while (!resp.ToUpper().Equals("Y"));

//Create output file

FileStream fs = File.Create(fileOutName);

fs.Close();

//Print new matrix to console

Matrix.WriteToFile(fileOutName, Matrix.ToString(nodeMatrix, nodeNames));

//Write Matrix to file

Console.WriteLine(Matrix.ToString(nodeMatrix, nodeNames));

Console.WriteLine("Was written to the given File.\n");

//Run program again?

Console.Write("Do you want to run again? [y/n]: ");

resp = Console.ReadLine();

} while (resp.ToUpper().Equals("Y"));

}

}

}