```
//
//  main.c
//  Mat3
//  A Method for Generating Security Assessment Criteria -
Coverage calculus
//
//  Created by Ferrucio de Franco Rosa on 12/09/15.
//  Copyright (c) 2015 Ferrucio de Franco Rosa. All rights
reserved.
//
// Last modified: 03/07/2016.
//

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/*
 Sequência Geral:
 readOntology(distanceFromDimension, distanceFromProperty,
identifiedBy);
 recebe listaDM, listaPP;
 cov_calc(CovDM, CovPP, CovLOC);
 retorna CovDM+CovPP+CovLOC;

 Simulation - Source 1 (SBIS-CFM NGS-1)

 ID  Descr       ListDM    ListPP         CovDM    CovPP
CovLOC
 1  NGS1 001    000100    11010010101    ?        ?        ?
 2  NGS1 002    110100    11010001010    ?        ?        ?
 3  NGS1 003    010010    10110010100    ?        ?        ?
 4  NGS1 004    100100    01001010101    ?        ?        ?
 5  NGS1 005    100100    11010001010    ?        ?        ?
 6  NGS1 006    100100    10101010100    ?        ?        ?
 7  NGS1 007    100100    11010001001    ?        ?        ?
 8  NGS1 008    100100    10011110100    ?        ?        ?
 9  NGS1 009    100100    11010001001    ?        ?        ?
 10 NGS1 010    111100    11010010011    ?        ?        ?
 11 NGS1 011    100100    10011110101    ?        ?        ?
 12 NGS1 012    101100    11010001001    ?        ?        ?
 13 NGS1 013    101101    11010010101    ?        ?        ?
 14 NGS1 014    100101    11011101001    ?        ?        ?
 15 NGS1 015    110111    11010001010    ?        ?        ?
 16 NGS1 016    111101    10101010101    ?        ?        ?
 17 NGS1 017    111100    11010001111    ?        ?        ?
```

```
 18 NGS1 018    010100    11010010011    ?         ?         ?
 19 NGS1 019    101100    10101010101    ?         ?         ?
 20 NGS1 020    100100    11010001111    ?         ?         ?
 21 NGS1 021    101001    11010001010    ?         ?         ?
 22 NGS1 022    100100    11010010101    ?         ?         ?
 23 NGS1 023    110000    10100101001    ?         ?         ?
 24 NGS1 024    100111    11010010011    ?         ?         ?
 25 NGS1 025    101100    10011110101    ?         ?         ?
 26 NGS1 026    101100    11010010100    ?         ?         ?
 27 NGS1 027    101100    10101001111    ?         ?         ?
 28 NGS1 028    001000    11010010100    ?         ?         ?
 29 NGS1 029    111100    11010010011    ?         ?         ?
 30 NGS1 030    111101    11110010101    ?         ?         ?
 31 NGS1 031    001000    11010010100    ?         ?         ?
 32 NGS1 032    101010    11010011100    ?         ?         ?
 33 NGS1 033    111101    11110001010    ?         ?         ?
 34 NGS1 034    101100    11010000010    ?         ?         ?
 35 NGS1 035    111100    11110000000    ?         ?         ?
 36 NGS1 036    111100    11010011111    ?         ?         ?
*/

float cov_calc(float A,int B)
{ float coverage=0.0;
    if (A == 0.0) return 0.0; // Min. Coverage = 0.0;
    else
      if ((coverage=A/B) >= 1.0) return 1.0; // Max.
Coverage = 1.0;
       else return coverage;
}


// Dimension Matrix
float MatDM[6][6] =
{
    {0.0, 0.5, 0.2, 0.6, 0.7, 0.9},
    {0.0, 0.0, 0.9, 0.7, 0.6, 0.8},
    {0.0, 0.0, 0.0, 0.4, 0.2, 0.6},
    {0.0, 0.0, 0.0, 0.0, 0.5, 0.2},
    {0.0, 0.0, 0.0, 0.0, 0.0, 0.8},
    {0.0, 0.0, 0.0, 0.0, 0.0, 0.0},
};
// Property Matrix
float MatPP[11][11] =
{
    {0.0, 0.9, 0.9, 0.9, 0.8, 0.8, 0.8, 0.8, 0.5, 0.2,
0.8},
```

```
    {0.0, 0.0, 0.9, 0.9, 0.8, 0.8, 0.8, 0.8, 0.5, 0.2,
0.2},
    {0.0, 0.0, 0.0, 0.9, 0.8, 0.8, 0.2, 0.8, 0.5, 0.8,
0.8},
    {0.0, 0.0, 0.0, 0.0, 0.2, 0.2, 0.8, 0.6, 0.5, 0.8,
0.4},
    {0.0, 0.0, 0.0, 0.0, 0.0, 0.2, 0.4, 0.6, 0.5, 0.8,
0.2},
    {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 0.2, 0.5, 0.8,
0.2},
    {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.8, 0.5, 0.2,
0.8},
    {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 0.2,
0.2},
    {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5,
0.5},
    {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.2},
    {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0},
};

// SumDist DM & PP
float sumDist(char *lista)
{
    int i, j;
    long len = strlen(lista);
    float SumDist=0.0;
    for (i=0;i<len;i++){//printf("Lista[%d]: %c \n", i+1,
lista[i]);
        if(lista[i] == '1'){ //printf("Lista[%d]: %c \n",
i+1, lista[i]);
            for (j=0;j<len;j++){
                if(len==6){
                    if (MatDM[i][j]>0.0 && lista[j] == '1')
{
                        printf("\n DistânciaDM(%d,%d): %.2f
\n", i+1, j+1, MatDM[i][j]);
                        SumDist = SumDist + MatDM[i][j];
                        // EdgeDM++;
                    }
                }
                else{
                    if (MatPP[i][j]>0.0 && lista[j] == '1')
{
                        printf("\n DistânciaPP(%d,%d): %.2f
```

```
\n", i+1, j+1, MatPP[i][j]);
                        SumDist = SumDist + MatPP[i][j];
                        // EdgePP++;
                    }
                }
            }
        }
    }
    return SumDist;
}

int main(int argc, char * argv[]) {
/*    ferruciof$ ./Mat3 "010011" "01110001001"
    0 Parametro: ./Mat3
    1 Parametro: 010011
    2 Parametro: 01110001001    */
    int MAX_DM = 6, MAX_PP = 11;
    char *listaDM, *listaPP;
    char coverage[19];
    float SumDistDM = 0.0, SumDistPP = 0.0;
    //    int VertexDM = 0, VertexPP = 0; /* Vertex (nós)
*/
    //    int EdgeDM = 0, EdgePP = 0; /* Edge = n*(n-1)/2
(arestas) */
    float CovDM=0.0, CovPP=0.0, CovLOC=0.0; /* Coverages */
    int cont;
    if (argc!=3) { printf("Número de parâmetros inválido");
exit (0);}
    for(cont=0; cont < argc; cont++)
        printf("%d Parametro: %s\n", cont,argv[cont]);

    listaDM = argv[1]; // DM list
    listaPP = argv[2]; // PP list

    // Calculate sum of distances DM & PP
    SumDistDM = sumDist(listaDM);
    SumDistPP = sumDist(listaPP);

    printf("ListaDM: %s  / ListaDM: %s  /  Tamanho DM: %lu
/  Tamanho PP: %lu\n", listaDM, listaPP, strlen(listaDM),
strlen(listaPP));

    // Calculate converage DM & PP
    CovDM = cov_calc(SumDistDM, MAX_DM);
    CovPP = cov_calc(SumDistPP, MAX_PP);
```

```
    // Calculate converage LOC
    CovLOC = cov_calc((CovDM+CovPP), 2);

    printf("\n\n SumDistDM: %.2f \n SumDistPP: %.2f \n",
SumDistDM, SumDistPP);
    printf("\n\n CovDM: %.3f \n CovPP: %.3f \n CovLOC: %.3f
\n", CovDM, CovPP, CovLOC );
    sprintf(coverage, "%.3f;%.3f;%.3f;", CovDM, CovPP,
CovLOC);
    printf("\n\n\nReturn Coverage from main():  %s \n\n\n",
coverage);
    return(0);
 }
```