

Joseph Morgan
Homework 4

CISP440

1 *Source Code*

```
1  #include <iostream>
2  #include <cstring>
3  #include <string>
4  #include <fstream>
5  #include <cstdlib>
6
7  #define MAX 40
8
9  using namespace std;
10
11 int R[MAX][MAX];    // a boolean array indicating members of a relation
12 int EC[MAX];        // a boolean array indicating representatives of equivalence
13                     classes
14 int size;
15
16 void printMatrix(int R[MAX][MAX])
17 {
18     int i, j;
19
20     for(i = 0; i < size; i++) {
21         for(j = 0; j < size; j++)
22             cout << R[i][j];
23         cout << endl;
24     }
25
26 int IsRefx(int R[MAX][MAX])
27 {
28     bool result = true;
29
30     for (int i = 0; i < size; ++i) {
31         if (R[i][i] != 1) {
32             result = false;
33         }
34     }
35
36     return result;
37 }
38
39 int IsSymt(int R[MAX][MAX])
40 {
41     bool result = true;
42
43     for (int i = 0; i < size; ++i) {
44         for (int j = 0; j < size; ++j) {
45             if (R[i][j] == 1 && R[j][i] != 1) {
46                 result = false;
47             }
48         }
49     }
50
51     return result;
52 }
53
```

```

54 void SquareMatrix(int R[MAX][MAX], int R2[MAX][MAX])
55 {
56     int temp = 0;
57
58     for (int i = 0; i < size; ++i) {
59         for (int j = 0; j < size; ++j) {
60             for (int k = 0; k < size; ++k) {
61                 temp += R[i][k] * R[j][k];
62             }
63
64             R2[i][j] = temp;
65             temp = 0;
66         }
67     }
68 }
69
70 int IsTrans(int R[MAX][MAX], int R2[MAX][MAX])
71 {
72     bool result = true;
73
74     SquareMatrix(R, R2);
75
76     for (int i = 0; i < size; ++i) {
77         for (int j = 0; j < size; ++j) {
78             if (R[i][j] == 0 && R2[i][j] != 0) {
79                 result = false;
80             }
81         }
82     }
83
84     return result;
85 }
86
87 void FindECs(int R[MAX][MAX], int EC[MAX])
88 {
89     for (int x = 0; x < MAX; ++x) EC[x] = 0;
90
91     EC[0] = 1;
92
93     for (int i = 1; i < size; ++i) {
94         EC[i] = 1;
95         for (int j = i - 1; j >= 0; --j) {
96             if (R[i][j]) EC[i] = 0;
97         }
98     }
99 }
100
101 void printECs(int R[MAX][MAX], int EC[MAX])
102 {
103     for (int i = 0; i < MAX; ++i) {
104         if (EC[i]) {
105             cout << "[" << i << "]_:_{";
106             for (int j = 0; j < size; ++j) {
107                 if (R[i][j]) cout << "_ " << j;
108             }
109             cout << "}_\n";
110         }

```

```

111     }
112
113     cout << endl;
114 }
115
116 int main()
117 {
118     char c;
119     char fnum[2];
120     char *start = "R";
121     char *end = ".bin";
122     for (int n = 1; n <= 7; ++n) {
123         char fname[7];
124         sprintf (fname, "%s%i%s", start, n, end);
125         ifstream fin (fname, ios_base::binary);
126         if ( !fin ) { cerr << "Input_file_could_not_be_opened\n"; exit(1); }
127
128         fin.read(&c, 1); size = c;
129
130         int i, j;
131         for(i = 0; i < size; i++)
132             for(j = 0; j < size; j++)
133             {
134                 fin.read(&c, 1);
135                 R[i][j] = c;
136             }
137
138         fin.close();
139
140         printMatrix(R);
141         cout << endl;
142
143         bool is_EQR = true;
144
145         IsRefx(R) ?
146             (cout << "Is_Reflexive\n") :
147             (is_EQR = false, cout << "Isn't_Reflexive\n");
148         IsSymt(R) ?
149             (cout << "Is_Symmetrical\n") :
150             (is_EQR = false, cout << "Isn't_Symmetrical\n");
151
152         int temp[MAX][MAX] = {{0}};
153         std::cout << "The_product_is:\n";
154         SquareMatrix(R, temp);
155         cout << endl;
156         printMatrix(temp);
157
158         int trans_test[MAX][MAX] = {{0}};
159         IsTrans(R, trans_test) ?
160             (cout << "Is_Transitive\n") :
161             (is_EQR = false, cout << "Isn't_Transitive\n");
162
163         if (is_EQR) {
164             FindECs(R, EC);
165             cout << "\nEquiv._classes:_\n";
166             printECs(R, EC);
167         }

```

168 }
169 }

2 *Output*

1101
1101
0010
1101

Is Reflexive
Is Symetrical
The product is:

3303
3303
0010
3303
Is Transitive

Equiv. classes:
[0] : { 0 1 3 }
[2] : { 2 }

1001
1111
0010
1101

Is Reflexive
Isn't Symetrical
The product is:

2202
2413
0110
2303
Isn't Transitive
1100
1100
0011
0011

Is Reflexive
Is Symetrical
The product is:

2200
2200
0022
0022
Is Transitive

Equiv. classes:
[0] : { 0 1 }
[2] : { 2 3 }

1100
1100
0001
0111

Isn't Reflexive
Isn't Symmetrical
The product is:

2201
2201
0011
1113
Isn't Transitive
110001
110001
001110
001110
001110
110001

Is Reflexive
Is Symmetrical
The product is:

330003
330003
003330
003330
003330
330003
Is Transitive

Equiv. classes:
[0] : { 0 1 5 }
[2] : { 2 3 4 }

111101
110001
001110
001110
011110
110101

Is Reflexive
Isn't Symmetrical
The product is:

532234
330013
203331
203331
313342
431124
Isn't Transitive
10001000100010000000000000000000


```

000000000000000000004000400040004
000000000000000000004000400040004000
00000000000000000000400040004000400
0000000000000000000040004000400040
000000000000000000004000400040004
000000000000000000004000400040004000
00000000000000000000400040004000400
0000000000000000000040004000400040
0000000000000000000040004000400040
000000000000000000004000400040004
Is Transitive

```

```

Equiv. classes:
[0] : { 0 4 8 12 }
[1] : { 1 5 9 13 }
[2] : { 2 6 10 14 }
[3] : { 3 7 11 }
[15] : { 15 19 23 27 }
[16] : { 16 20 24 28 }
[17] : { 17 21 25 29 }
[18] : { 18 22 26 30 }

```