Joseph Morgan
Extra Credit

CISP440

1 *Source Code*

```
// CISP440 Extra Credit
// by Joseph Morgan
//
// Originally, this was written using classes and object-oriented design, with
// which I'm much more comfortable than procedural C. I decided to modify this
// program to compile with no warnings with the --std=c99 flag, as well as
// -Wall, -pedantic, etc. in order to be sure to 100% within the assignment
// specifications. Professor Ross mentioned at one point that he wants
// assignments in this class to be written in C, without using features
// implemented exclusively in C++. Some of the provided code for other
// assignemnts uses C++ classes (iostream and fstream, for example), but I
// wanted to err on the side of caution. Also, it was more fun to do something
// I'm less familiar with.
//
// I included some functions mostly as they were written in the last homework
// assignment for testing purposes.
//
// The algorithm I used is:
//
//  1. Create an empty logical matrix for the relation to be generated.
//
//  2. Create an empty logical matrix to store equiv. classes to be generated.
//
//  3. For each row r in the relation matrix,
//
//    a. Check each of your equiv. classes for a row with r. For example,
//         if you're looking at matrix[3][x], you need to add matrix[3][3] to
//         that row. You don't want to add the 3 if it already exists in an
//         equiv. class, so you scan through all classes to make sure it's not
//         there.
//
//      i. If an equiv. class exists with column r, copy that class to row r
//           of the relation matrix. Now you've got matrix[r][r] to satisfy
//           reflexivity, and you haven't broken transitivity by overlapping an
//           equiv. class.
//
//      ii. If no equiv. class exists with r, randomly generate one, being
//           careful to make sure you do not ovelap any values already in
//           an equiv. class. Manually insert r value. Copy to relation matrix.
//
// I'm not sure if this is a particularly elegant solution, but I feel like
// I was successful in avoiding a 'guess-and-check' method.
//
// There are various command-line arguments that can be provided to influence
// the execution or output of generator. See the manpage for details.
// Just kidding.
//
// Use [-e density] to pick how densely the matrix will be populated. Should
// probably keep it from 0 - 10, I'm not sure what will happen if you don't.
//
// Use [-s size] to pick how many columns x rows the matrix will be. Values
// should be between 2 and 40.
//
// Use [-v] to make the generator output the set of relations in a list,
```

```c
// the equiv. classes in the relation, etc. By default, the only thing that
// gets output is the matrix itself.
//
// With no arguments, the generator will pseudo-randomly pick a size and
// density. Go with this option if you like to live on the edge.

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

static const int MAX = 40;
static int debug = 0;
static int verbose = 0;
static int g_size = 0;
static int g_density = 0;

struct Relation
{
  int **matrix;
  int size;
  int *equiv_classes;
};
typedef struct Relation Relation;

Relation* init_relation_blank (int size);
Relation* init_relation_from_file (const char *filename);
void free_relation (Relation *r);
void print_matrix (Relation *r);
void print_relation (Relation *r);
int is_reflexive (Relation *r);
int is_symetric (Relation *r);
void square_matrix (Relation *r, Relation *s);
int is_transitive (Relation *r);
void find_equiv_classes (Relation *r);
void print_equiv_classes (Relation *r);
Relation* gen_random_relation();
int overlaps (int matrix[MAX][MAX], int x, int y);

int main (int argc, char *argv[])
{
  srand ((unsigned)time(NULL));
  for (int i = 1; i < argc; ++i) {
    if (strcmp (argv[i], "-s") == 0) {
      g_size = atoi(argv[i++ + 1]);
    } else if (strcmp (argv[i], "-e") == 0) {
      g_density = atoi (argv[i++ + 1]);
    } else if (strcmp (argv[i], "-d") == 0) {
      debug = 1;
    } else if (strcmp (argv[i], "-v") == 0) {
      verbose = 1;
    } else if (argc != 1) {
      fprintf
        (stderr, "Usage: %s [-s size] [-e density] [-v] [-d] [-h]\n", argv[0]);
      fprintf
        (stderr, "    -s size: Size of the matrix to be randomly generated\n");
      fprintf
```

```c
        (stderr, "    -e density: Likelyhood that any given relation will exist. Higher means less likely\n");
      fprintf
        (stderr, "    -v: enable verbose output\n");
      fprintf
        (stderr, "    -d: enable debugging output\n");
      fprintf
        (stderr, "    -h: print this message\n");
      exit(1);
    }
  }

  Relation *test_rand = gen_random_relation();
  printf("Matrix: \n");
  print_matrix (test_rand);
  if (verbose) {
    printf("\n\nSet of Relations: \n");
    print_relation (test_rand);
    printf("\n\nEquivalence Classes: \n");
    find_equiv_classes (test_rand);
    print_equiv_classes (test_rand);
    int is_EQR = 1;
    printf("\n");
    is_reflexive(test_rand) ?
      (printf("Is Reflexive\n")) :
      (is_EQR = 0, printf("Isn't Reflexive\n"));
    is_symetric(test_rand) ?
      (printf("Is Symetrical\n")) :
      (is_EQR = 0, printf("Isn't Symetrical\n"));
    is_transitive(test_rand) ?
      (printf("Is Transitive\n")) :
      (is_EQR = 0, printf("Isn't Transitive\n"));
  }
  return 0;
}

Relation* init_relation_blank (int size)
{
  // Create 2d array
  Relation* new_relation = (Relation*)malloc(sizeof(Relation));
  new_relation->matrix = malloc(MAX * sizeof(int*));
  for (unsigned int i = 0; i < MAX; ++i)
    new_relation->matrix[i] = malloc(MAX * sizeof(int));

  // Zero out array
  for (unsigned int i = 0; i < MAX; ++i) {
    for (unsigned int j = 0; j < MAX; ++j) {
      new_relation->matrix[i][j] = 0;
    }
  }
  new_relation->size = size;

  // Init equiv classes
  new_relation->equiv_classes = malloc(MAX * sizeof(int));
  for (unsigned int i = 0; i < MAX; ++i)
    new_relation->equiv_classes[i] = 0;
  return new_relation;
```

```
}

void free_relation (Relation* r)
{
  for (unsigned int i = 0; i < MAX; ++i)
    free (r->matrix[i]);
  free (r->matrix);
  free (r->equiv_classes);
  free (r);
}

Relation* init_relation_from_file (const char *filename)
{
  Relation *new_relation;

  // Read file for matrix values
  char c;
  FILE *infile;
  infile = fopen(filename, "rb");
  if (!infile) {
    fprintf(stderr, "Input file could not be opened\n");
    exit(1);
  }
  fscanf(infile, "%c", &c);
  new_relation = init_relation_blank ((int)c);
  for (int i = 0; i < new_relation->size; i++) {
    for (int j = 0; j < new_relation->size; j++) {
      fscanf(infile, "%c", &c);
      new_relation->matrix[i][j] = c;
    }
  }
  return new_relation;
}

void print_matrix (Relation *r)
{
  for (int i = 0; i < r->size; i++) {
    for (int j = 0; j < r->size; j++) {
      printf(" %i", r->matrix[i][j]);
    }
    printf("\n");
  }
}

void print_relation (Relation *r)
{
  printf("R = {\n");
  for (int i = 0; i < r->size; ++i) {
    for (int j = 0; j < r->size; ++j) {
      if (r->matrix[i][j])
        printf(" (%i, %i)", i, j);
    }
    printf("\n");
  }
  printf("}\n");
}
```

```c
int is_reflexive (Relation *r)
{
  int result = 1;

  for (int i = 0; i < r->size; ++i)
    if (r->matrix[i][i] != 1)
      result = 0;

  return result;
}

int is_symetric (Relation *r)
{
  int result = 1;

  for (int i = 0; i < r->size; ++i)
    for (int j = 0; j < r->size; ++j)
      if ((r->matrix[i][j] == 1 && r->matrix[j][i] != 1) ||
          (r->matrix[j][i] == 1 && r->matrix[i][j] != 1))
        result = 0;
  return result;
}

void square_matrix (Relation *r, Relation *s)
{
  int temp = 0;

  for (int i = 0; i < r->size; ++i) {
    for (int j = 0; j < r->size; ++j) {
      for (int k = 0; k < r->size; ++k) {
        temp += r->matrix[i][k] * r->matrix[j][k];
      }
      s->matrix[i][j] = temp;
      temp = 0;
    }
  }
}

int is_transitive (Relation *r)
{
  int result = 1;
  Relation* squared_r = init_relation_blank(r->size);
  square_matrix(r, squared_r);

  for (int i = 0; i < r->size; ++i) {
    for (int j = 0; j < r->size; ++j) {
      if (r->matrix[i][j] == 0 && squared_r->matrix[i][j] != 0) {
        result = 0;
      }
    }
  }
  free_relation (squared_r);
  return result;
}

void find_equiv_classes (Relation* r)
{
```

```
  r->equiv_classes[0] = 1;

  for (int i = 1; i < r->size; ++i) {
    r->equiv_classes[i] = 1;
    for (int j = i - 1; j >= 0; --j) {
      if (r->matrix[i][j]) r->equiv_classes[i] = 0;
    }
  }
}

void print_equiv_classes (Relation* r)
{
  for (int i = 0; i < MAX; ++i) {
    if (r->equiv_classes[i]) {
      printf("[%i] : {", i);
      for (int j = 0; j < r->size; ++j) {
        if (r->matrix[i][j]) printf(" %i", j);
      }
      printf(" } \n");
    }
  }
  printf("\n");
}

Relation* gen_random_relation()
{
  int size;
  if (g_size)
    size = g_size;
  else
    size = (rand() % (MAX - 3)) + 4;
  unsigned int density;
  if (g_density)
    density = g_density;
  else
    density = (rand() % 4) + 5;
  int eq_rel_found = 0;
  int ec_matrix[MAX][MAX];
  for (int i = 0; i < MAX; ++i)
    for (int j = 0; j < MAX; ++j)
      ec_matrix[i][j] = 0;
  int ec_matrix_row= 0;
  Relation* rand_relation = init_relation_blank (size);

  ec_matrix[0][0] = 1;
  for (int i = 0; i < size; ++i)
    if (rand() % 10 >= density)
      ec_matrix[0][i] = 1;
  ++ec_matrix_row;

  for (int r = 0; r < size; ++r) {
    if (debug) printf("Attempting to write row %i\n", r);
    for (int p = 0; !eq_rel_found && p <= r ; ++p) {
      if (debug) printf("Scanning ec_matrix row %i\n", p);
      if (ec_matrix[p][r]) {
        if (debug) printf("(x, %i) is is row %i of ec_matrix\n", r, p);
        for (int c = 0; c < size; ++c) {
```

```
            rand_relation->matrix[r][c] = ec_matrix[p][c];
            if(debug) {
              printf("Copying %i from ec_matrix[%i, %i]", ec_matrix[p][c], p, c);
              printf(" to rand_relation->matrix[%i, %i]\n", r, c);
            }
          }
          eq_rel_found = 1;
        }
      }
      if (!eq_rel_found) {
        ec_matrix[ec_matrix_row][r] = 1;
        for (int y = 1; y < size; ++y)
          if (rand() % 10 >= density && !overlaps(ec_matrix, ec_matrix_row, y))
            ec_matrix[ec_matrix_row][y] = 1;
        for (int c = 0; c < size; ++c) {
          rand_relation->matrix[r][c] = ec_matrix[ec_matrix_row][c];
        }
        ++ec_matrix_row;
      }
      eq_rel_found = 0;
    }
    return rand_relation;
}

int overlaps (int matrix[MAX][MAX], int x, int y)
{
    int result = 0;
    for (int i = 0; i < x; ++i)
      if (matrix[i][y])
        result = 1;
    return result;
}
```

2 *Output*

```
Size = 4, Density = Random
Matrix:
 1 0 1 0
 0 1 0 1
 1 0 1 0
 0 1 0 1


Set of Relations:
R = {
 (0, 0) (0, 2)
 (1, 1) (1, 3)
 (2, 0) (2, 2)
 (3, 1) (3, 3)
}


Equivalence Classes:
[0] : { 0 2 }
[1] : { 1 3 }
```

```
Is Reflexive
Is Symetrical
Is Transitive


Size = 5, Density = Random
***
Matrix:
 1 0 1 0 0
 0 1 0 0 0
 1 0 1 0 0
 0 0 0 1 0
 0 0 0 0 1


Set of Relations:
R = {
 (0, 0) (0, 2)
 (1, 1)
 (2, 0) (2, 2)
 (3, 3)
 (4, 4)
}


Equivalence Classes:
[0] : { 0 2 }
[1] : { 1 }
[3] : { 3 }
[4] : { 4 }


Is Reflexive
Is Symetrical
Is Transitive


Size = 6, Density = Random
***
Matrix:
 1 0 1 0 1 0
 0 1 0 0 0 0
 1 0 1 0 1 0
 0 0 0 1 0 0
 1 0 1 0 1 0
 0 0 0 0 0 1


Set of Relations:
R = {
 (0, 0) (0, 2) (0, 4)
 (1, 1)
 (2, 0) (2, 2) (2, 4)
 (3, 3)
 (4, 0) (4, 2) (4, 4)
```

```
 (5, 5)
}


Equivalence Classes:
[0] : { 0 2 4 }
[1] : { 1 }
[3] : { 3 }
[5] : { 5 }


Is Reflexive
Is Symetrical
Is Transitive


Size = 7, Density = Random
***
Matrix:
 1 0 0 0 0 1 1
 0 1 1 1 0 0 0
 0 1 1 1 0 0 0
 0 1 1 1 0 0 0
 0 0 0 0 1 0 0
 1 0 0 0 0 1 1
 1 0 0 0 0 1 1


Set of Relations:
R = {
 (0, 0) (0, 5) (0, 6)
 (1, 1) (1, 2) (1, 3)
 (2, 1) (2, 2) (2, 3)
 (3, 1) (3, 2) (3, 3)
 (4, 4)
 (5, 0) (5, 5) (5, 6)
 (6, 0) (6, 5) (6, 6)
}


Equivalence Classes:
[0] : { 0 5 6 }
[1] : { 1 2 3 }
[4] : { 4 }


Is Reflexive
Is Symetrical
Is Transitive


Size = 8, Density = Random
***
Matrix:
 1 1 0 0 0 0 0 0
 1 1 0 0 0 0 0 0
 0 0 1 0 0 0 0 1
```

```
 0 0 0 1 0 0 0 0
 0 0 0 0 1 0 1 0
 0 0 0 0 0 1 0 0
 0 0 0 0 1 0 1 0
 0 0 1 0 0 0 0 1


Set of Relations:
R = {
  (0, 0) (0, 1)
  (1, 0) (1, 1)
  (2, 2) (2, 7)
  (3, 3)
  (4, 4) (4, 6)
  (5, 5)
  (6, 4) (6, 6)
  (7, 2) (7, 7)
}


Equivalence Classes:
[0] : { 0 1 }
[2] : { 2 7 }
[3] : { 3 }
[4] : { 4 6 }
[5] : { 5 }


Is Reflexive
Is Symetrical
Is Transitive


Size = 9, Density = Random
***
Matrix:
 1 0 0 1 0 0 0 0 0
 0 1 0 0 1 0 0 0 0
 0 0 1 0 0 0 0 1 0
 1 0 0 1 0 0 0 0 0
 0 1 0 0 1 0 0 0 0
 0 0 0 0 0 1 0 0 0
 0 0 0 0 0 0 1 0 0
 0 0 1 0 0 0 0 1 0
 0 0 0 0 0 0 0 0 1


Set of Relations:
R = {
  (0, 0) (0, 3)
  (1, 1) (1, 4)
  (2, 2) (2, 7)
  (3, 0) (3, 3)
  (4, 1) (4, 4)
  (5, 5)
  (6, 6)
  (7, 2) (7, 7)
```

```
 (8, 8)
}


Equivalence Classes:
[0] : { 0 3 }
[1] : { 1 4 }
[2] : { 2 7 }
[5] : { 5 }
[6] : { 6 }
[8] : { 8 }


Is Reflexive
Is Symetrical
Is Transitive


Size = 10, Density = Random
***
Matrix:
 1 1 0 0 1 0 0 1 0 0
 1 1 0 0 1 0 0 1 0 0
 0 0 1 1 0 0 1 0 1 1
 0 0 1 1 0 0 1 0 1 1
 1 1 0 0 1 0 0 1 0 0
 0 0 0 0 0 1 0 0 0 0
 0 0 1 1 0 0 1 0 1 1
 1 1 0 0 1 0 0 1 0 0
 0 0 1 1 0 0 1 0 1 1
 0 0 1 1 0 0 1 0 1 1


Set of Relations:
R = {
 (0, 0) (0, 1) (0, 4) (0, 7)
 (1, 0) (1, 1) (1, 4) (1, 7)
 (2, 2) (2, 3) (2, 6) (2, 8) (2, 9)
 (3, 2) (3, 3) (3, 6) (3, 8) (3, 9)
 (4, 0) (4, 1) (4, 4) (4, 7)
 (5, 5)
 (6, 2) (6, 3) (6, 6) (6, 8) (6, 9)
 (7, 0) (7, 1) (7, 4) (7, 7)
 (8, 2) (8, 3) (8, 6) (8, 8) (8, 9)
 (9, 2) (9, 3) (9, 6) (9, 8) (9, 9)
}


Equivalence Classes:
[0] : { 0 1 4 7 }
[2] : { 2 3 6 8 9 }
[5] : { 5 }


Is Reflexive
Is Symetrical
Is Transitive
```

```
Size = 15, Density = Random
***
Matrix:
 1 1 1 0 0 1 1 0 0 0 0 0 1 0 1
 1 1 1 0 0 1 1 0 0 0 0 0 1 0 1
 1 1 1 0 0 1 1 0 0 0 0 0 1 0 1
 0 0 0 1 0 0 0 1 0 1 1 0 0 0 0
 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0
 1 1 1 0 0 1 1 0 0 0 0 0 1 0 1
 1 1 1 0 0 1 1 0 0 0 0 0 1 0 1
 0 0 0 1 0 0 0 1 0 1 1 0 0 0 0
 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
 0 0 0 1 0 0 0 1 0 1 1 0 0 0 0
 0 0 0 1 0 0 0 1 0 1 1 0 0 0 0
 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0
 1 1 1 0 0 1 1 0 0 0 0 0 1 0 1
 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0
 1 1 1 0 0 1 1 0 0 0 0 0 1 0 1


Set of Relations:
R = {
 (0, 0) (0, 1) (0, 2) (0, 5) (0, 6) (0, 12) (0, 14)
 (1, 0) (1, 1) (1, 2) (1, 5) (1, 6) (1, 12) (1, 14)
 (2, 0) (2, 1) (2, 2) (2, 5) (2, 6) (2, 12) (2, 14)
 (3, 3) (3, 7) (3, 9) (3, 10)
 (4, 4) (4, 11) (4, 13)
 (5, 0) (5, 1) (5, 2) (5, 5) (5, 6) (5, 12) (5, 14)
 (6, 0) (6, 1) (6, 2) (6, 5) (6, 6) (6, 12) (6, 14)
 (7, 3) (7, 7) (7, 9) (7, 10)
 (8, 8)
 (9, 3) (9, 7) (9, 9) (9, 10)
 (10, 3) (10, 7) (10, 9) (10, 10)
 (11, 4) (11, 11) (11, 13)
 (12, 0) (12, 1) (12, 2) (12, 5) (12, 6) (12, 12) (12, 14)
 (13, 4) (13, 11) (13, 13)
 (14, 0) (14, 1) (14, 2) (14, 5) (14, 6) (14, 12) (14, 14)
}


Equivalence Classes:
[0] : { 0 1 2 5 6 12 14 }
[3] : { 3 7 9 10 }
[4] : { 4 11 13 }
[8] : { 8 }


Is Reflexive
Is Symetrical
Is Transitive


Size = 20, Density = Random
***
Matrix:
```

```
 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0
 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0
 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0
 0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0
 0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0
 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0
 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1


Set of Relations:
R = {
 (0, 0) (0, 3) (0, 17)
 (1, 1) (1, 2)
 (2, 1) (2, 2)
 (3, 0) (3, 3) (3, 17)
 (4, 4) (4, 5) (4, 15)
 (5, 4) (5, 5) (5, 15)
 (6, 6) (6, 11) (6, 16)
 (7, 7) (7, 8) (7, 9) (7, 12)
 (8, 7) (8, 8) (8, 9) (8, 12)
 (9, 7) (9, 8) (9, 9) (9, 12)
 (10, 10)
 (11, 6) (11, 11) (11, 16)
 (12, 7) (12, 8) (12, 9) (12, 12)
 (13, 13)
 (14, 14)
 (15, 4) (15, 5) (15, 15)
 (16, 6) (16, 11) (16, 16)
 (17, 0) (17, 3) (17, 17)
 (18, 18)
 (19, 19)
}


Equivalence Classes:
[0] : { 0 3 17 }
[1] : { 1 2 }
[4] : { 4 5 15 }
[6] : { 6 11 16 }
[7] : { 7 8 9 12 }
[10] : { 10 }
[13] : { 13 }
[14] : { 14 }
[18] : { 18 }
```

```
[19] : { 19 }


Is Reflexive
Is Symetrical
Is Transitive


Size = 10, Density = 9
Matrix:
 1 0 0 0 0 1 0 0 0 0
 0 1 0 1 0 0 0 0 0 0
 0 0 1 0 0 0 0 0 0 0
 0 1 0 1 0 0 0 0 0 0
 0 0 0 0 1 0 0 1 0 0
 1 0 0 0 0 1 0 0 0 0
 0 0 0 0 0 0 1 0 0 0
 0 0 0 0 1 0 0 1 0 0
 0 0 0 0 0 0 0 0 1 1
 0 0 0 0 0 0 0 0 1 1


Set of Relations:
R = {
 (0, 0) (0, 5)
 (1, 1) (1, 3)
 (2, 2)
 (3, 1) (3, 3)
 (4, 4) (4, 7)
 (5, 0) (5, 5)
 (6, 6)
 (7, 4) (7, 7)
 (8, 8) (8, 9)
 (9, 8) (9, 9)
}


Equivalence Classes:
[0] : { 0 5 }
[1] : { 1 3 }
[2] : { 2 }
[4] : { 4 7 }
[6] : { 6 }
[8] : { 8 9 }


Is Reflexive
Is Symetrical
Is Transitive


Size = 10, Density = 5
Matrix:
 1 1 1 1 0 0 1 0 1 1
 1 1 1 1 0 0 1 0 1 1
 1 1 1 1 0 0 1 0 1 1
 1 1 1 1 0 0 1 0 1 1
```

```
 0 0 0 0 1 0 0 1 0 0
 0 0 0 0 0 1 0 0 0 0
 1 1 1 1 0 0 1 0 1 1
 0 0 0 0 1 0 0 1 0 0
 1 1 1 1 0 0 1 0 1 1
 1 1 1 1 0 0 1 0 1 1


Set of Relations:
R = {
 (0, 0) (0, 1) (0, 2) (0, 3) (0, 6) (0, 8) (0, 9)
 (1, 0) (1, 1) (1, 2) (1, 3) (1, 6) (1, 8) (1, 9)
 (2, 0) (2, 1) (2, 2) (2, 3) (2, 6) (2, 8) (2, 9)
 (3, 0) (3, 1) (3, 2) (3, 3) (3, 6) (3, 8) (3, 9)
 (4, 4) (4, 7)
 (5, 5)
 (6, 0) (6, 1) (6, 2) (6, 3) (6, 6) (6, 8) (6, 9)
 (7, 4) (7, 7)
 (8, 0) (8, 1) (8, 2) (8, 3) (8, 6) (8, 8) (8, 9)
 (9, 0) (9, 1) (9, 2) (9, 3) (9, 6) (9, 8) (9, 9)
}


Equivalence Classes:
[0] : { 0 1 2 3 6 8 9 }
[4] : { 4 7 }
[5] : { 5 }


Is Reflexive
Is Symetrical
Is Transitive


Size = 10, Density = 1
Matrix:
 1 1 1 1 1 1 1 0 0 1
 1 1 1 1 1 1 1 0 0 1
 1 1 1 1 1 1 1 0 0 1
 1 1 1 1 1 1 1 0 0 1
 1 1 1 1 1 1 1 0 0 1
 1 1 1 1 1 1 1 0 0 1
 1 1 1 1 1 1 1 0 0 1
 0 0 0 0 0 0 0 1 1 0
 0 0 0 0 0 0 0 1 1 0
 1 1 1 1 1 1 1 0 0 1


Set of Relations:
R = {
 (0, 0) (0, 1) (0, 2) (0, 3) (0, 4) (0, 5) (0, 6) (0, 9)
 (1, 0) (1, 1) (1, 2) (1, 3) (1, 4) (1, 5) (1, 6) (1, 9)
 (2, 0) (2, 1) (2, 2) (2, 3) (2, 4) (2, 5) (2, 6) (2, 9)
 (3, 0) (3, 1) (3, 2) (3, 3) (3, 4) (3, 5) (3, 6) (3, 9)
 (4, 0) (4, 1) (4, 2) (4, 3) (4, 4) (4, 5) (4, 6) (4, 9)
 (5, 0) (5, 1) (5, 2) (5, 3) (5, 4) (5, 5) (5, 6) (5, 9)
 (6, 0) (6, 1) (6, 2) (6, 3) (6, 4) (6, 5) (6, 6) (6, 9)
```

```
 (7, 7) (7, 8)
 (8, 7) (8, 8)
 (9, 0) (9, 1) (9, 2) (9, 3) (9, 4) (9, 5) (9, 6) (9, 9)
}


Equivalence Classes:
[0] : { 0 1 2 3 4 5 6 9 }
[7] : { 7 8 }


Is Reflexive
Is Symetrical
Is Transitive


Size = Random, Density = Random
***
Matrix:
 1 0 0 0 0 0 1 0 0 1 1 0 1 1 1 0 0 1 1 0 1 1 1
 0 1 1 0 1 1 0 0 1 0 0 1 0 0 0 1 1 0 0 1 0 0 0
 0 1 1 0 1 1 0 0 1 0 0 1 0 0 0 1 1 0 0 1 0 0 0
 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 1 1 0 1 1 0 0 1 0 0 1 0 0 0 1 1 0 0 1 0 0 0
 0 1 1 0 1 1 0 0 1 0 0 1 0 0 0 1 1 0 0 1 0 0 0
 1 0 0 0 0 0 1 0 0 1 1 0 1 1 1 0 0 1 1 0 1 1 1
 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 1 1 0 1 1 0 0 1 0 0 1 0 0 0 1 1 0 0 1 0 0 0
 1 0 0 0 0 0 1 0 0 1 1 0 1 1 1 0 0 1 1 0 1 1 1
 1 0 0 0 0 0 1 0 0 1 1 0 1 1 1 0 0 1 1 0 1 1 1
 0 1 1 0 1 1 0 0 1 0 0 1 0 0 0 1 1 0 0 1 0 0 0
 1 0 0 0 0 0 1 0 0 1 1 0 1 1 1 0 0 1 1 0 1 1 1
 1 0 0 0 0 0 1 0 0 1 1 0 1 1 1 0 0 1 1 0 1 1 1
 1 0 0 0 0 0 1 0 0 1 1 0 1 1 1 0 0 1 1 0 1 1 1
 0 1 1 0 1 1 0 0 1 0 0 1 0 0 0 1 1 0 0 1 0 0 0
 0 1 1 0 1 1 0 0 1 0 0 1 0 0 0 1 1 0 0 1 0 0 0
 1 0 0 0 0 0 1 0 0 1 1 0 1 1 1 0 0 1 1 0 1 1 1
 1 0 0 0 0 0 1 0 0 1 1 0 1 1 1 0 0 1 1 0 1 1 1
 0 1 1 0 1 1 0 0 1 0 0 1 0 0 0 1 1 0 0 1 0 0 0
 1 0 0 0 0 0 1 0 0 1 1 0 1 1 1 0 0 1 1 0 1 1 1
 1 0 0 0 0 0 1 0 0 1 1 0 1 1 1 0 0 1 1 0 1 1 1
 1 0 0 0 0 0 1 0 0 1 1 0 1 1 1 0 0 1 1 0 1 1 1


Set of Relations:
R = {
 (0, 0) (0, 6) (0, 9) (0, 10) (0, 12) (0, 13) (0, 14) (0, 17) (0, 18) (0, 20) (0,
    21) (0, 22)
 (1, 1) (1, 2) (1, 4) (1, 5) (1, 8) (1, 11) (1, 15) (1, 16) (1, 19)
 (2, 1) (2, 2) (2, 4) (2, 5) (2, 8) (2, 11) (2, 15) (2, 16) (2, 19)
 (3, 3)
 (4, 1) (4, 2) (4, 4) (4, 5) (4, 8) (4, 11) (4, 15) (4, 16) (4, 19)
 (5, 1) (5, 2) (5, 4) (5, 5) (5, 8) (5, 11) (5, 15) (5, 16) (5, 19)
 (6, 0) (6, 6) (6, 9) (6, 10) (6, 12) (6, 13) (6, 14) (6, 17) (6, 18) (6, 20) (6,
    21) (6, 22)
 (7, 7)
 (8, 1) (8, 2) (8, 4) (8, 5) (8, 8) (8, 11) (8, 15) (8, 16) (8, 19)
```

```
 (9, 0) (9, 6) (9, 9) (9, 10) (9, 12) (9, 13) (9, 14) (9, 17) (9, 18) (9, 20) (9,
    21) (9, 22)
 (10, 0) (10, 6) (10, 9) (10, 10) (10, 12) (10, 13) (10, 14) (10, 17) (10, 18) (10,
    20) (10, 21) (10, 22)
 (11, 1) (11, 2) (11, 4) (11, 5) (11, 8) (11, 11) (11, 15) (11, 16) (11, 19)
 (12, 0) (12, 6) (12, 9) (12, 10) (12, 12) (12, 13) (12, 14) (12, 17) (12, 18) (12,
    20) (12, 21) (12, 22)
 (13, 0) (13, 6) (13, 9) (13, 10) (13, 12) (13, 13) (13, 14) (13, 17) (13, 18) (13,
    20) (13, 21) (13, 22)
 (14, 0) (14, 6) (14, 9) (14, 10) (14, 12) (14, 13) (14, 14) (14, 17) (14, 18) (14,
    20) (14, 21) (14, 22)
 (15, 1) (15, 2) (15, 4) (15, 5) (15, 8) (15, 11) (15, 15) (15, 16) (15, 19)
 (16, 1) (16, 2) (16, 4) (16, 5) (16, 8) (16, 11) (16, 15) (16, 16) (16, 19)
 (17, 0) (17, 6) (17, 9) (17, 10) (17, 12) (17, 13) (17, 14) (17, 17) (17, 18) (17,
    20) (17, 21) (17, 22)
 (18, 0) (18, 6) (18, 9) (18, 10) (18, 12) (18, 13) (18, 14) (18, 17) (18, 18) (18,
    20) (18, 21) (18, 22)
 (19, 1) (19, 2) (19, 4) (19, 5) (19, 8) (19, 11) (19, 15) (19, 16) (19, 19)
 (20, 0) (20, 6) (20, 9) (20, 10) (20, 12) (20, 13) (20, 14) (20, 17) (20, 18) (20,
    20) (20, 21) (20, 22)
 (21, 0) (21, 6) (21, 9) (21, 10) (21, 12) (21, 13) (21, 14) (21, 17) (21, 18) (21,
    20) (21, 21) (21, 22)
 (22, 0) (22, 6) (22, 9) (22, 10) (22, 12) (22, 13) (22, 14) (22, 17) (22, 18) (22,
    20) (22, 21) (22, 22)
}


Equivalence Classes:
[0] : { 0 6 9 10 12 13 14 17 18 20 21 22 }
[1] : { 1 2 4 5 8 11 15 16 19 }
[3] : { 3 }
[7] : { 7 }


Is Reflexive
Is Symetrical
Is Transitive


Size = Random, Density = Random
***
Matrix:
 1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 0 1 1 0 0 1 0 1 0 0 0 0 1 0
 0 1 1 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
 0 1 1 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1
 0 1 1 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1
 1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 0 1 1 0 0 1 0 1 0 0 0 0 1 0
 1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 0 1 1 0 0 1 0 1 0 0 0 0 1 0
 0 1 1 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1
 0 1 1 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
 1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 0 1 1 0 0 1 0 1 0 0 0 0 1 0
 1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 0 1 1 0 0 1 0 1 0 0 0 0 1 0
 1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 0 1 1 0 0 1 0 1 0 0 0 0 1 0
```

```
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1
1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 0 1 1 0 0 1 0 1 0 0 0 0 1 0
1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 0 1 1 0 0 1 0 1 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0
1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 0 1 1 0 0 1 0 1 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 0 1 1 0 0 1 0 1 0 0 0 0 1 0
0 1 1 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0
1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 0 1 1 0 0 1 0 1 0 0 0 0 1 0
0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1


Set of Relations:
R = {
 (0, 0) (0, 7) (0, 8) (0, 12) (0, 13) (0, 14) (0, 17) (0, 18) (0, 21) (0, 23) (0,
    28)
 (1, 1) (1, 2) (1, 4) (1, 9) (1, 11) (1, 24)
 (2, 1) (2, 2) (2, 4) (2, 9) (2, 11) (2, 24)
 (3, 3) (3, 6) (3, 10) (3, 16) (3, 25) (3, 29)
 (4, 1) (4, 2) (4, 4) (4, 9) (4, 11) (4, 24)
 (5, 5) (5, 15)
 (6, 3) (6, 6) (6, 10) (6, 16) (6, 25) (6, 29)
 (7, 0) (7, 7) (7, 8) (7, 12) (7, 13) (7, 14) (7, 17) (7, 18) (7, 21) (7, 23) (7,
    28)
 (8, 0) (8, 7) (8, 8) (8, 12) (8, 13) (8, 14) (8, 17) (8, 18) (8, 21) (8, 23) (8,
    28)
 (9, 1) (9, 2) (9, 4) (9, 9) (9, 11) (9, 24)
 (10, 3) (10, 6) (10, 10) (10, 16) (10, 25) (10, 29)
 (11, 1) (11, 2) (11, 4) (11, 9) (11, 11) (11, 24)
 (12, 0) (12, 7) (12, 8) (12, 12) (12, 13) (12, 14) (12, 17) (12, 18) (12, 21) (12,
    23) (12, 28)
 (13, 0) (13, 7) (13, 8) (13, 12) (13, 13) (13, 14) (13, 17) (13, 18) (13, 21) (13,
    23) (13, 28)
 (14, 0) (14, 7) (14, 8) (14, 12) (14, 13) (14, 14) (14, 17) (14, 18) (14, 21) (14,
    23) (14, 28)
 (15, 5) (15, 15)
 (16, 3) (16, 6) (16, 10) (16, 16) (16, 25) (16, 29)
 (17, 0) (17, 7) (17, 8) (17, 12) (17, 13) (17, 14) (17, 17) (17, 18) (17, 21) (17,
    23) (17, 28)
 (18, 0) (18, 7) (18, 8) (18, 12) (18, 13) (18, 14) (18, 17) (18, 18) (18, 21) (18,
    23) (18, 28)
 (19, 19)
 (20, 20) (20, 27)
 (21, 0) (21, 7) (21, 8) (21, 12) (21, 13) (21, 14) (21, 17) (21, 18) (21, 21) (21,
    23) (21, 28)
 (22, 22)
 (23, 0) (23, 7) (23, 8) (23, 12) (23, 13) (23, 14) (23, 17) (23, 18) (23, 21) (23,
    23) (23, 28)
 (24, 1) (24, 2) (24, 4) (24, 9) (24, 11) (24, 24)
 (25, 3) (25, 6) (25, 10) (25, 16) (25, 25) (25, 29)
 (26, 26)
 (27, 20) (27, 27)
```

(28, 0) (28, 7) (28, 8) (28, 12) (28, 13) (28, 14) (28, 17) (28, 18) (28, 21) (28, 23) (28, 28)
(29, 3) (29, 6) (29, 10) (29, 16) (29, 25) (29, 29)
}

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

Is Reflexive
Is Symetrical
Is Transitive

Size = Random, Density = Random
***
Matrix:

1	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0	1
0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0
0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0
1	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0	1
0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0	1
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0
0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
1	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0	1
0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0	1
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0	1
1	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0	1
1	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0	1
0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0
0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0
1	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0	1

Set of Relations:
R = {
(0, 0) (0, 3) (0, 7) (0, 13) (0, 17) (0, 19) (0, 20) (0, 21) (0, 25)
(1, 1) (1, 2) (1, 9) (1, 15) (1, 22) (1, 24)

```
 (2, 1) (2, 2) (2, 9) (2, 15) (2, 22) (2, 24)
 (3, 0) (3, 3) (3, 7) (3, 13) (3, 17) (3, 19) (3, 20) (3, 21) (3, 25)
 (4, 4) (4, 12) (4, 23)
 (5, 5) (5, 14) (5, 16)
 (6, 6) (6, 10)
 (7, 0) (7, 3) (7, 7) (7, 13) (7, 17) (7, 19) (7, 20) (7, 21) (7, 25)
 (8, 8) (8, 18)
 (9, 1) (9, 2) (9, 9) (9, 15) (9, 22) (9, 24)
 (10, 6) (10, 10)
 (11, 11)
 (12, 4) (12, 12) (12, 23)
 (13, 0) (13, 3) (13, 7) (13, 13) (13, 17) (13, 19) (13, 20) (13, 21) (13, 25)
 (14, 5) (14, 14) (14, 16)
 (15, 1) (15, 2) (15, 9) (15, 15) (15, 22) (15, 24)
 (16, 5) (16, 14) (16, 16)
 (17, 0) (17, 3) (17, 7) (17, 13) (17, 17) (17, 19) (17, 20) (17, 21) (17, 25)
 (18, 8) (18, 18)
 (19, 0) (19, 3) (19, 7) (19, 13) (19, 17) (19, 19) (19, 20) (19, 21) (19, 25)
 (20, 0) (20, 3) (20, 7) (20, 13) (20, 17) (20, 19) (20, 20) (20, 21) (20, 25)
 (21, 0) (21, 3) (21, 7) (21, 13) (21, 17) (21, 19) (21, 20) (21, 21) (21, 25)
 (22, 1) (22, 2) (22, 9) (22, 15) (22, 22) (22, 24)
 (23, 4) (23, 12) (23, 23)
 (24, 1) (24, 2) (24, 9) (24, 15) (24, 22) (24, 24)
 (25, 0) (25, 3) (25, 7) (25, 13) (25, 17) (25, 19) (25, 20) (25, 21) (25, 25)
}


Equivalence Classes:
[0] : { 0 3 7 13 17 19 20 21 25 }
[1] : { 1 2 9 15 22 24 }
[4] : { 4 12 23 }
[5] : { 5 14 16 }
[6] : { 6 10 }
[8] : { 8 18 }
[11] : { 11 }


Is Reflexive
Is Symetrical
Is Transitive
```