**CS 238** **Discrete Mathematics II**

# Closeness

*Relations* *Exploration*

## Goal

The purpose of this exploration is for you to explore and enhance an implementation of a representation of relations to discover their basic properties.

## Requirements

Write a C++ program that takes inputs which are files containing connection (zero-one) matrices of binary relations of a set with itself. Determine which properties each relation has.

Start with the stub code supplied. Add your code and submit the file with the *same name* (`relations.cpp`).

If conditions are right, you can build and test your code in the Linux Lab via the command:

```
make it just so
```

## Grading Criteria

The rubric below is meant to guide you in your quest for exceptional quality.

| | **Exceptional**<br>**100%** | **Good**<br>**90%** | **Acceptable**<br>**70%** | **Developing**<br>**50%** | **Missing**<br>**0%** |
|---|---|---|---|---|---|
| **Correctness/ Completeness**<br>**50%** | Code compiles and runs correctly for all input files (as verified by **make it just so**). | Code compiles and runs correctly for at least 90% of the input files. | Code compiles and runs correctly for at least 70% of the input files. | Code compiles, but gives 50% erroneous output (more or less). | Nothing correct, code either missing altogether, or does not compile. |
| **Elegance**<br>**50%** | **Good**, plus eliminated **all** redundant code. | **Acceptable**, plus overloaded another appropriate operator for use in the **isTransitive** test, plus eliminated some redundant code. | Overloaded the '*' operator for computing the Boolean product. | Used encapsulation but not operator overloading. But at least the code runs correctly, as first and foremost, an elegant solution is a correct solution. | Used neither encapsulation nor operator overloading, and what's worse, the submitted code doesn't run correctly. |