Assignment-3 –The SetCalculator
COMPSYS 202 / MECHENG 270

Partha Roop

1 Introduction

Discrete structures such as **sets** are foundational for the design of any system involving a computer. Hence, they are the main basis of **discrete maths**, a discipline of maths that is essential for the formalization of concepts used in computer engineering, mechatronics, and computer science.

While a set is a collection of unique members, there can be a relation that satisfies among them. For example, we have a set containing three numbers 1,2, and 3. Considering the "greater than" relation, then we have a set of relation "2 is greater than 1", "3 is greater than 2", and "3 is greater than 1". As such, a relation on a set is essentially again a set. The set of relations naturally is the inheritance relationship in the object oriented programming (OOP).

The **objective** of this assignment is to develop a program called a set calculator. While the program is running, the user can open the input file, which contains the information about a set of strings and a set of relations. After extracting the information, the set calculator program checks whether the relation is reflexive, symmetric, and/or transitive. Furthermore, in case of equivalence relation, the equivalence class is computed.

1.1 Learning outcomes

1. Discrete Structures such as *Sets, Relations, and Equivalence Classes* as object-oriented concepts.
2. Using the above mathematical entities to learn about OO concepts: encapsulation, inheritance and composition.
3. Improve the understanding of unit testing and test driven development learnt through the lectures and also the labs in week 10, 11.
4. Practice the use of `vectors` and `iterators` and develop confidence in string operations.

2 Getting started

The SetCalculator program has normal mode and unit testing mode. In normal mode, the user can dynamically interface with the program. Whereas, in testing mode, only the unit testing result is printed on the terminal.

2.0.1 Normal Mode

1. To get the skeleton code source files for this assignment, you must accept the GitHub Classroom invitation that will be shared with you. This will then clone the code in the Release folder into a private GitHub repository for you to use. **This repository must remain private during the lifetime of the assignment**. When you accept the GitHub assignment, you can clone the repository either to your own computer or work on it via Repl. For the latter, a Repl badge/image is automatically added to the README.md file when you accept the GitHub invitation. By clicking on this, you will be able to run your code in the online Repl IDE.

2. The `run` command in repl.it can be configured in the .replit file. It has been setup with two configurations corresponding to the Normal and Unit testing mode respectively. These must be enabled in an XOR fashion.

3. Next, enter `./calculator.o` at the repl.it command prompt to run the program.

4. While the program is running, enter `open TestCases/a.txt`. This will open a input file in the TestCases folder.

5. Enter `list`. It will show the extracted information in the input file.

6. You can type `check -r` to check if this relation is reflexive. However, currently this function is left unimplemented for this assignment. Also, try `check -s`, `check -t`, and `check -e`.

7. There is also `eqclass` command. type *help* to know more about the commands.

8. Finally, type `exit` to terminate the program.

2.0.2 Unit testing Mode

1. In the .replit file comment out the `run` command for Normal Mode and uncomment the equivalent `run` command for the Unit Testing mode(for the unit testing, the program is compiled with a special flag *debug*).

2. Enter `./calculator.o` to run the program. You will see the unit testing result.

3. Open SetControl.cpp file using any text editor. At the bottom of this code, there is *Testing* function, where you can manage the unit testing scenario.

2.1 Frequent Submissions (Github)

As you work on your assignment, you must make frequent git commits. If you only commit your final code, it will look suspicious and you be penalised. In the case of this assignment, you should aim to commit your changes to GitHub every time you implement new functionality (e.g. a method). You can check your commits in your GitHub account to make sure the frequent commits are being recorded. **Using GitHub to frequently commit your changes is mandatory in this course**. In addition to teaching you to use a useful software development skill (version control with git), it will also protect you in two very important ways:

1. should something terrible happen to your laptop, or your files get corrupted, or you submitted the wrong files to Canvas, or you submitted late, etc, then you are protected as the commits in GitHub verify the progress in your assignment (i.e. what you did and when).

2. If your final code submission looks suspiciously similar to someone else (see academic honesty section below), then GitHub provides a track record demonstrating how you progressed the assignment during that period. Together, GitHub is there to help you.

3 Background Information

In this section, we describe the background knowledge needed to complete this assignment.

3.1 Binary Relations

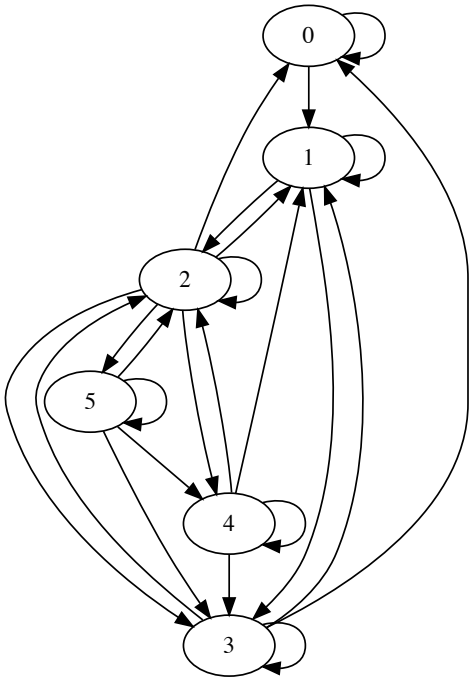
A relation, by definition, is a subset of the product of sets i.e., $R \subseteq S_1 \times S_2 \times .. \times S_n$. Here $S_1, .., S_n$ are the participating sets and any subset of the product set is a relation, by definition. The above relation is known as an n-ary relation.

In this assignment, we will be only interested in one specific type of relation called a **binary relation**. Here, $R \subseteq S \times S$. Such relations have many practical applications, such as say in your GPS navigator. We will elaborate the further on this, using the concept of **graphs**, in the following section.

3.2 Input File Format and Visualization

In the input file, the information about the set and relation is stored as a graph. Graphs are mathematical objects of the form $< V, E >$ where V denotes a set of vertices and E denotes a set of edges. Graphs are used in multitude of applications. A very prominent one is is GPS navigation, where the destinations in a city, for example could be represented as the vertices and the edges between them represent the different paths.

In our example, we will consider simple graphs. Consider the example graph shown in Figure 1. Note that the set of vertices V can be considered as the set using which a relation may be created. In this example $V = \{0, 1, 2, 3, 4, 5\}$. The relation captures the the edges of the graph. In this example the adjacency relation between vertices may be captured as a relation $R = \{(0, 0), (0, 1), (1, 1), (1, 2), (1, 3), (2, 0), (2, 1), (2, 2), (2, 3), (2, 4), (2, 5), (3, 3), (3, 0), (3, 1), (3, 2), (4, 2), (4, 3), (4, 1), (4, 4), (5, 5), (5, 4), (5, 3), (5, 2)\}$. Such a relation can be also represented using a specific text format in Figure 1b. Using a popular tool for graphs called Graphviz, a graph is visualized as in Figure 1. There is an web application for Graphviz tool at `www.webgraphviz.com`.



(a) Graph visualization.

```
// 0, 1, 2, 3, 4, 5
digraph testgraph{
0 -> 0;
0 -> 1;
1 -> 1;
1 -> 2;
1 -> 3;
2 -> 0;
2 -> 1;
2 -> 2;
2 -> 3;
2 -> 4;
2 -> 5;
3 -> 3;
3 -> 0;
3 -> 1;
3 -> 2;
4 -> 2;
4 -> 3;
4 -> 1;
4 -> 4;
5 -> 5;
5 -> 4;
5 -> 3;
5 -> 2;
}
```

(b) Text representation.

Figure 1: An example graph of a set and relation.

The input file can be visualized for manual validation of your SetCalculator program. Try to visualize the sample input files in the TestCases folder. This can be done simply by copying the entire text in the file to the Graphviz tool.

It is important that the first line in Figure 1b must capture all the nodes in the graph. This is clear because that the relation are described only using the existing nodes. However, it is possible that the input file is invalid. For your assignment, *you can assume that only the correct input files are tested during the marking process.*

4 Submission

In this section, we clearly indicate your tasks, and guide how to submit your assignment successfully.

4.1 Your Tasks

You are required to implement two commands: *check* and *eqclass*. The logic for these commands should be programmed in the SetOfStrings.cpp and StringRelation.cpp files only. This means that, you should not modify the provided Makefile, main.cpp, SetControl.cpp/h, and SetUI.cpp/h files. However, for unit testing, you can modify the *Testing* function in the SetControl.cpp.

Remember that the StringRelation class inherits the SetOfStrings class. Also, the StringRelation class defines a SetOfStrings object called SetMembers as an attribute. This also indicates that they also have a composition relationship. Exploit the inheritance and composition relationship to program your logic. Note that the SetMembers object stores the nodes information (see the first line in Figure 1b).

4.2 How to submit your code

This assignment deadline is **5 pm 23 October 2019**. You must submit a zip file, which contains:

1. "Code" folder: this folder contains your SetOfStrings.cpp/h and StringsRelation.cpp/h files. Do not include other files. Any cpp/h files other than SetOfStrings and StringsRelation will be ignored during the marking process.

2. Coversheet: this should include your full name, id, upi, and your declaration of originality form.

We will be testing your code on Repl. Please double check your submission can be compiled and run on Repl, because if your code fails to compile, then you will automatically lose almost all the marks.

4.3 Academic Honesty

- The work done on this assignment must be your own work. Think carefully about any problems you come across, and try to solve them yourself before you ask anyone for help. Under no circumstances should you take or pay for an electronic copy of someone else's work.
- All submitted code will be checked using software similarity tools. Code detected for suspicious similarity will result in an Investigative Meeting and will be forwarded to the Disciplinary Committee.
- Penalties for copying will be severe – to avoid being caught copying, don't do it.
- To ensure you are not identified as cheating you should follow these points:
  - Always do individual assignments by yourself.
  - Never give another person your code.
  - Never put your code in a public place (e.g., forum, git, your website).
  - Never leave your computer unattended. You are responsible for the security of your account.
  - Ensure you always remove your USB flash drive from the computer before you log off.
  - Ensure that you make regular commits to Github reflecting your changes.

4.4 Late submissions

Late submissions incur the following penalties:

- 15% penalty for zero to 24 hours late
- 30% penalty for 25 to 48 hours late
- 100% penalty for over 48 hours late (dropbox automatically closes)

You must double check that you have uploaded the correct code for marking! There will be no exceptions if you accidentally submitted the wrong files, regardless if you can prove you did not modify them since the deadline. No exceptions.