SQE — Assignment 2: Control-Flow Graph

1. Introduction

In this assignment, you will analyze a simple function using CFG and find the minimal set of paths for covering the function behavior.

2. Goals

To practice CFG, including generating the graph and finding the minimal sets for the statement, branch, and predicate criteria.

3. cfg.jar

We will evaluate your assignment using the *cfg.jar* that is included on the assignment page. This file can be used both for practicing the CFG material and for evaluating the answers.

To run this file, you will need Java 17 or higher. If you want to generate PDF files from the generated graph, you can install Graphviz (not obligatory).

To run the jar file, write the following command in the shell:

```
java -jar cfg.jar
```

This command will output the possible commands of the tool — generate, extract, and evaluate:

```
Usage: java -jar cfg.jar [-hV] COMMAND
Executes the CFG tool
  -h, --help      Show this help message and exit.
  -V, --version   Print version information and exit.
Commands:
  generate  Generate code from a seed or a control-flow graph from code.
  extract   Extract the minimal paths given a control-flow graph or a code file.
  evaluate  Evaluate the graph or the minimal paths.

See 'java -jar cfg.jar <command> -h' for more information on a specific command.
```

As you can see, the program's menu is self-explanatory and you can get further details on each command by running: `java -jar cfg.jar <command-name>`.

To generate a pdf from the dot file, run: `dot -Tpdf graph.dot -o graph.pdf`

Alternatively, you can see the graph by pasting it in Graphviz Online.

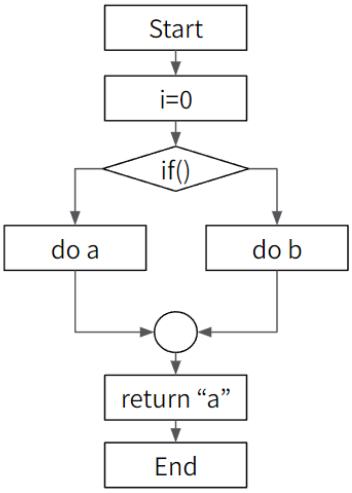
4. The Assignment task

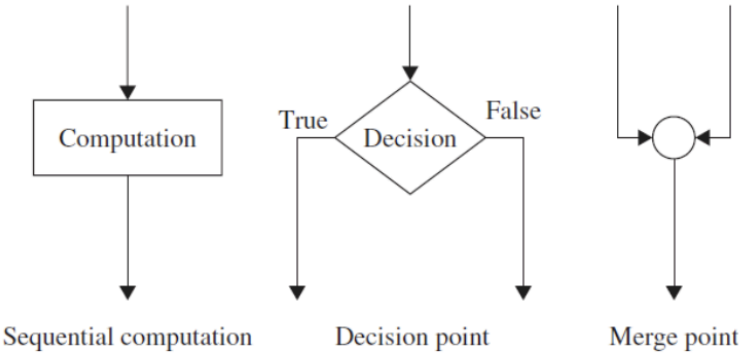
1. Use seed `40` and level EASY to generate a function.
2. Calculate for this function, the control-flow graph and a minimal path set for statement, branch, and predicate coverage.
3. Submit the graph and the sets.

4. Submission Constraints

Deliverables: Submit two files: `graph.txt` and `paths.json`

`graph.txt`: This file will contain the CFG of the given function in the dot language. Please note that the graph should impose the graph rules we specified in class:





All edges must have a label "T" or "F".

NOTE: we will not check the syntax (i.e., that computation nodes are boxed, decision nodes are diamond and that there is a merge point). Nevertheless, we urge you to get used to this type of writing since we will check it in the exam.

See an example of such a file in the appendix.

`paths.json`:

A JSON file in the format of:

```
{
  "STATEMENT": [
    "1-2-3-4-5-6"
  ],
  "BRANCH": [
    "1-2-3-4-5-6",
    "1-2-3-4-6"
  ]
 }
```

5. Deadline

The files should be submitted through Moodle by **04/02/2024 23:59**.

GOOD LUCK

Appendix

Code sample:

```
public class CodeSample {
  public void run(int textId) {
    String text = null;
    if(textId < 0) {
      text = "smaller";
    } else if(textId > 0) {
      text = "bigger";
    } else {
      text = null;
    }

    if(text == null) {
      System.out.println("Can't find text with id. Returning
new default text");
      text = "equals";
    }
    System.out.println("The text is: " + text);
  }
}
```

Expected format of `graph.txt` file:

```
strict digraph G {
  1 [ label="1: Start" shape="box" ];
  2 [ label="2: init textId" shape="box" ];
  3 [ label="3: text = null" shape="box" ];
  4 [ label="4: textId < 0" shape="diamond" ];
  5 [ label="5: text = 'smaller'" shape="box" ];
  6 [ label="6: textId > 0" shape="diamond" ];
  7 [ label="7: text = 'bigger'" shape="box" ];
  9 [ label="9: text = null" shape="box" ];
  10 [ shape="circle" ];
  12 [ label="12: text == null" shape="diamond" ];
  13 [ label="13: sout(...)" shape="box" ];
  14 [ label="14: text = 'equals'" shape="box" ];
  15 [ shape="circle" ];
  16 [ label="16: sout(...)" shape="box" ];
  17 [ label="17: End" shape="box" ];
  1 -> 2 [ label="T" ];
  2 -> 3 [ label="T" ];
  3 -> 4 [ label="T" ];
  4 -> 5 [ label="T" ];
  4 -> 6 [ label="F" ];
  5 -> 10 [ label="T" ];
  6 -> 7 [ label="T" ];
  6 -> 9 [ label="F" ];
  9 -> 10 [ label="T" ];
  7 -> 10 [ label="T" ];
  10 -> 12 [ label="T" ];
  12 -> 13 [ label="T" ];
  12 -> 15 [ label="F" ];
  13 -> 14 [ label="T" ];
  14 -> 15 [ label="T" ];
  15 -> 16 [ label="T" ];
  16 -> 17 [ label="T" ];
}
```

The graph will look like this: