Project 1: `basher`

Summer of Programming 2021
Department of Computer Science
University of Copenhagen

July 15, 2021

1 Introduction

In Computer Systems, you will be working a little bit with *shell scripting*, which provides one of many ways of setting up a testing framework.

For this project, you will be writing a program `basher`, which generates a simple shell script from a given configuration file.

2 Configuration file layout

The configuration file is a *comma separated values* file. Each row has four columns:

- **Tested program**: A path to the executable program that is meant to be tested.
- **Input type**: The type of input given to the program. This is one of three types:
  - `STDIN` - The input should be written to the command-line interface after the program is started.
  - `ARGS` - The input should be given verbatim as an argument to the program when started from the command line.
  - `FILE` - The input should be written to a file, after which the name of the file must be provided as the sole argument to the program.

- **Input data**: The input the program receives, formatted depending on the input type.
- **Expected output**: The output expected to be printed by the program.

For example, a configuration file for testing the program with the relative path `programs/regex` may have rows such as:

```
programs/regex,FILE,ab ab,Found match
programs/regex,FILE,a|b a,Found match
programs/regex,FILE,(a*)b ba,Could not find match
```

3 `basher` specification

The `basher` program should take a single file `f` as the command line argument. It should then generate an output file `test.sh`, which is structured in the following manner:

- At the start, create two temporary files for storing the program output:

  ```
  ACTUAL=$(mktemp)
  EXPECTED=$(mktemp)
  ```

- For each row in the configuration file:
  - If the row has input type `STDIN`, write the following into `test.sh`:

    ```
    echo "{R}" > $EXPECTED
    echo "{D}" | ./{P} > $ACTUAL
    if ! diff -q $ACTUAL $EXPECTED
    then
      echo "Failed test of {P} with standard input {D}"
    fi
    ```

  - If the row has input type `ARGS`, write the following into `test.sh`:

```
echo "{R}" > $EXPECTED
./{P} "{D}" > $ACTUAL
if ! diff -q $ACTUAL $EXPECTED
then
  echo "Failed test of {P} with argument {D}"
fi
```

- If the row has input type `FILE`, write the following into `test.sh`:

```
TMPFILE=$(mktemp)
echo "{D}" > $TMPFILE
echo "{R}" > $EXPECTED
./{P} $TMPFILE > $ACTUAL
if ! diff -q $ACTUAL $EXPECTED
then
  echo "Failed test of {P} with input file content {D}"
fi
```

where `{P}` is the program, `{D}` is the input data, and `{R}` is the expected output, all specified by the row.

3.1 Verification

If you want to test that your program is working as intended, you can try and write some tests for the bugged calculator `calc.c`, also found in this directory, and see if you can find the bug. You can compile an executable `calc` using `gcc`:

```
gcc calc.c -o calc
```

For guidance on how to use the calculator, use `calc -h`.