Homework. 2: Core C++Language

Ignacio Vizzo, E-Mail ivizzo@uni-bonn.de

Handout : 30.04.2020
Handin: 15.05.2020 at 23:59:59 (CET)

General rules

1. You need to provide the build system for this homework. This means, you need to provide as many **CMakeLists.txt** files as you think is needed.
2. ALL targets should be placed inside a 'bin' folder. This time there is no need to create a results' directory. This means that the binaries should be placed in `cpp-homeworks/homework_2/task_<n>/bin/`
3. Your code will be checked using `clang-format`. If your code in not propperly formatted, then the tests will not run.
4. Your code will be static-analyzed by `clang-tidy`, if your code does not comply with the analysis, the tests will not run. If you are using VSCode with the reccomended setup, most of the errors will be visible while you program.
5. You will be provided with the tests that the automated checker is running, if you want to run these tests on your own you will need to install `bats`
6. A `cmake` folder is also provided with some helper modules that you could use while developing your homeworks. These modules includes
   - clang-format
   - clang-tidy
   - cppcheck

A First C++ program: Guessing Game (5 points)

1. **(5 points)** In this exercise you will write a small "guessing" game. The program will pick a random number from 0 to 99 and you will be guessing this number by providing your guess through `stdin`. Depending on which number you have guessed there are 3 outcomes:
   - You have guessed the number. Then the program has to tell you that you have won.
   - Your number if larger than the target one. The program should tell you that it's number is smaller.
   - Your number is smaller than the target one. The program should tell you that its number is larger.

A.1 Requirements to pass the tests

- The game goes on until you have guessed the number.
- The name of the output program should be `task_1`
- The program should return an error code(1) and print this exact string in case a failure is encountered: **"Error encountered, exiting..."**
- The program should print a warning msg to the `stderr` in case the user provides an invalid input and using this exact string: **"[WARNING] : Number must be between 0 and 99"**
- The last output that the program **must** inform to the user must always contain the genreated random numnber. It doesn't matter if it's under a failure case or a success guess.
- The generated random number should be **random**. The tests will check that you don't provide always the same number. Make sure you test your program locally a few times before submission.

A.2 Tips

Tip:

(a) To generate a random number you can use a `random_device`: `https://en.cppreference.com/w/cpp/numeric/random/random_device`

(b) You could use `std::cin.fail()` to check for errors in the `stdin`

(c) The `<cstdlib>` header defines two macros that can help you to report failure or sucess of your program: `EXIT_SUCCESS, EXIT_FAILURE`

B Input Parameters and streams (5 points)

2. **(5 points)** The program will take as input **only** 2 arguments. Both arguments will be filenames with the following format: "<INT>.<EXT>", examples of input arguments are shown below. Your program will need to process these input arguments and provide some information about them, all the outputs should be printed to the `stdout`.

B.1 Requirements to pass the tests

- The name of the output program should be `task_2`
- You should check that only two and just two input arguments are provided to your program, otherwise print an error msg and abort the execution.
- If the files are with **".txt"** extension, you need to output the mean value between the first and last number provided in the arguments.
- If the files are with **".png"** extension, you need to output the sum of the values between the first and last number provided in the arguments.
- If the first file extension is **".txt"** and the last one **".png"** you should output the modulo division between the first one and the last one
- If any of the extensions doesn't match the above conditions, then you must abort the program execution and print an error msg to the `stderr`

NOTE: In case of sucess(no error), the output should be within the last line of whatever you print to `stdout`, otherwise tests will fail.

B.2 Usage Example

```
$ cd cpp-homeworks/homework_2
$ ./bin/task_2 100.txt 200.txt
$ ./bin/task_2 100.doc 100.doc # Invalid should give an error
```