CS-4100/5100
PJ03 (Corona version) TEST FILES

xjc – The XJack Compiler

The Test Suite for PJ03 consists of eight Jack programs, each in its own directory. Two of these are standard Jack Programs and the others contain specific elements of the XJack language.

Each (X)Jack program consists of two classes – 'Main.jack' and 'Tests.jack'. There is also a 'Test.tst' test file for the CPU Emulator that you will need to run.

With the exception of the two standard Jack programs ('standard' and 'errors'), the Test.tst file does NOT compare the results to a compare file; instead, it runs to completion putting all of the output to the 'test.out' file (which you will need to submit).

The 'standard' program tests against the provided 'test.cmp' file and it intended to allow you to make sure that the process you are using works. The 'errors' program has twenty errors in the 'Main.jack' file and also has a 'test.cmp' file that the test file compares against. More about this program later.

The process of running the tests is straightforward.

1. Copy your OS files (either the .jack or the .vm files) into each folder.
2. Compile the .jack files in the folder.
3. Translate the .vm files in the folder.
4. Run the CPU Emulator using the Test.tst script.

Do not forget to do Step 1 – it's rather important!

The 'errors' program has twenty errors to test your compiler's ability to identify and report them. It is not expected that your compiler will identify all twenty on the first attempt. This is the approach that is recommended.

Run your compiler on the program and capture the report it makes regarding errors. For each error caught, copy the comment located in Main.jack into a text file and below it paste the relevant output from your compiler. If capturing text is not practical, then do whatever works, but please submit only a single file containing all of the output, be it a .pdf or a .docx or whatever.

Once you have caught a particular error, fix the error (the correct code is contained in the comment on the same line as the error).

Once your compiler is not catching any more errors, if there are still errors in the file it is possible that fixing some of them will enable the compiler to catch some of the others. So, starting with the first remaining error in the file, copy the comment to the text file and below it put “UNCAUGHT ERROR”. Then fix the error and recompile. Do this one error at a time to give your compiler the best chance at catching the greatest number of errors. There are a few errors in the program that it is expected that your compiler will not catch (possible extra credit if it does).

Continue this process until all of the errors are fixed (at which point your program should pass the test script using the provided compare file).

SUBMISSION

As in previous projects, you will submit a single .zip named “pj03_username.zip” where username is replaced with your UCCS username. The zip file should contain a single folder (at the top level) named “pj03_username” (with your username). In that folder should be a folder named “code” and a folder named “test”.

In the “code” folder you need to place the source code for you compiler. You may use whatever folder structure below this that is most convenient.

In the “test” folder you should have the eight folders for the eight (X)Jack programs. In addition to the original files in each folder, you need to have your OS .vm files (plus the OS .jack files if there are any), the ‘Main.vm’ and ‘Tests.vm’ files, the final .asm file, the ‘test.out’ file, and any other files (such as symbol tables, error output, etc.) produced by your tools. You might also add a ‘readme.txt’ file containing any information you would like the grader to take into account.

NOTE: In the Project Assignment document it states that you do not need to submit the results of running your program on the CPU Emulator. That has changed because some of the test files take a bit of time to run and we want to streamline the grading process. But the grader will randomly select programs to run both with the VM Emulator and the CPU Emulator, so be sure that the output files you submit are consistent with the code you generated.

Remember, if you used someone else’s tools in any way, be sure to acknowledge them appropriately.