Assignment 1 Description

Weighting and Due Date

- Marks for this assignment contribute 5% of the overall course mark.
- Marks for functionality will be awarded automatically by the web submission system.
- **Due dates: Milestone** - **11:55pm Tuesday of week 7**, **Final** - **11:55pm Friday of week 7**.
- **Late penalties:** For each part, the maximum mark awarded will be reduced by 25% per day / part day late. If your mark is greater than the maximum, it will be reduced to the maximum.
- **Core Body of Knowledge (CBOK) Areas:** abstraction, design, hardware and software, data and information, and programming.

Project Description

In this assignment you will implement a tokeniser that with minor changes could be used to complete variations of projects 6, 7, 8, 10 and 11 in the nand2tetris course. A detailed description of the requirements are shown below. The exectuable programs, **tokens** and **tokens-context** will read text from standard input and produce a list of tokens in the text on standard output.

Note: you should complete workshop 05 before attempting this assignment.

SVN Repository

Note: this assignment assumes that you have already created directories for every assignment, workshop, project and exam in your svn repository, as described on the **Startup Files for Workshops and Assignments (https://myuni.adelaide.edu.au/courses/54311/pages/startup-files-for-workshops-and-assignments)** page.

1. If required, checkout a working copy of the assignment1 directory from your svn repository.
2. Change directory to the working copy of the assignment1 directory.
3. Copy the newest zip file attached below into the updates sub-directory and add it to svn.
4. Run the following command to place the assignment's startup files in the correct locations:

```
% make install
```

5. Add the tokensier.cpp file and an empty tests directory to your svn repository:

```
% svn add tokeniser.cpp tokeniser-basics.cpp
% svn add --depth=empty tests
% svn commit -m "Assignment 1 Startup Files"
```

Assignment 1 Files and Directories

In addition to the generic **Makefile** and **updates** sub-directory, the assignment1 directory should now contain the following files and directories:

- **tokens** - executable script that will run your compiled **tokens** program.
- **tokens.cpp** C++ source file containing the **main()** function for **tokens.**
- **tokens-context** - executable script that will run your compiled **tokens-context** program.
- **tokens-context.cpp** C++ source file containing the **main()** function for **tokens-context.**
- **tokeniser.cpp** C++ source file containing the **next_token()** function.
- **tokeniser-basics.cpp** C++ source file containing input functions.
- **bin** - this directory contains precompiled programs and scripts.
- **includes** - this directory contains **.h** files for the library.
- **lib** - this directory contains precompiled library components.
- **originals** - this directory contains the original version of the **tokeniser.cpp**.
- **tests** - this directory contains test data, you can add your own tests here.

Note: you need to edit the **tokeniser.cpp** and **tokeniser-basics.cpp** files to complete this assignment. All the other files are automatically regenerated every time you run **make**, they must not be changed or added to **svn**. You can add extra test inputs to the **tests** directory but those are the only additional files that you may add to **svn**.

Note: if a newer version of the startup files is made available, it must be placed in the **updates** sub-directory and added to **svn**. The next time **make** is run, all of the files will be updated except for **tokeniser.cpp**.

Submission and Marking Scheme

Submissions for this assignment must be made to the **web submission system** (https://cs.adelaide.edu.au/services/websubmission) *assignment* named: ***Assignment 1 - Submit Here***. The assessment is based on "**Assessment of Programming Assignments** (https://myuni.adelaide.edu.au/courses/54311/pages/assessment-of-programming-assignments) ".

Note: the ***Submit Here*** assignment will show a breakdown of your marks by category but it will always show your total mark as capped at **0**.

Your tokeniser program must be written in C++. Your **tokens** and **tokens-context** programs will be compiled using the **Makefile** and the **tokens.cpp** or **tokens-context.cpp** files included in the zip file attached below together with the **tokeniser.cpp** and **tokeniser-basics.cpp** files in your svn directory. Your programs will then be tested using the set of test files that are attached below. In addition a number of ***secret*** tests will also be run. **Note**: if your program fails any of these ***secret*** tests you **will not** receive any feedback about these ***secret*** tests, even if you ask!

Assignment 1 - Milestone Submissions: due 11:55pm Tuesday of week 7

The marks awarded by the **web submission system** (https://cs.adelaide.edu.au/services/websubmission) for the milestone submission contribute up to 20% of your marks for assignment 1. The marks for the Milestone Tests are used as the marks for the milestone submission. The Milestone Tests test the milestone token definitions shown below using your **tokens** program.
Your milestone submission mark, after the application of late penalties, will be posted to the myuni gradebook when the assignment marking is complete.

You can view the Milestone Tests marks in the **Milestone** *assignment* but submissions must be made using the ***Assignment 1 - Submit Here*** *assignment.*

Assignment 1 - Final Submissions: due 11:55pm Friday of week 7

The marks awarded for the final submission contribute up to 80% of your marks for assignment 1.

Your final submission mark will be the geometric mean of three components, the marks for the Final Tests, a mark for your logbook and a mark for your code. It will be limited to 20% more than the marks for the Final Tests. See "**Assessment - Mark Calculations** (https://myuni.adelaide.edu.au/courses/54311/pages/assessment-mark-calculations) " for examples of how the marks are combined. The Final Tests are all tests run, including those used for the Milestone Tests. Your final submission mark, after the application of late penalties, will be posted to the myuni gradebook when the assignment marking is complete.

NOTE - A logbook mark of 0 results in a Final Submission mark of 0.

You can view the Final Tests marks in the **Final** *assignment* but submissions must be made using the ***Assignment 1 - Submit Here*** *assignment.*

Logbook Marking

Important: the logbook must have entries for all work in this assignment, including your milestone submissions. See "**Assessment - Logbook Review (https://myuni.adelaide.edu.au/courses/54311/pages/assessment-logbook-review)** " for details of how your logbook will be assessed.

Code Review Marking

For each of your programming assignments you are expected to submit well written **code**. See "**Assessment - Code Review (https://myuni.adelaide.edu.au/courses/54311/pages/assessment-code-review)** " for details of how your code will be assessed.

Assignment 1 - Participation Marks

Any submissions to assignment 1 that are made before the due date for Milestone Submissions may be awarded up to 10 participation marks. The participation marks will be the marks awarded for the Final Tests divided by 10. You can view the participation marks awarded in the ***Pre Milestone*** *assignment* but submissions must be made using the ***Assignment 1 - Submit Here*** *assignment*. The participation marks will be allocated to week 6.

Tokenisers

Background

The primary task of any language translator is to work out how the structure and meaning of an input in a given language so that an appropriate translation can be output in another language. If you think of this in terms of a natural language such as English. When you attempt to read a sentence you do not spend your time worrying about what characters there are, how much space is between the letters or where lines are broken. What you do is consider the words and attempt to derive structure and meaning from their order and arrangement into English language sentences, paragraphs, sections, chapters etc. In the same way, when we attempt to write translators from assembly language, virtual machine language or a programming language into another form, we attempt to focus on things like keywords, identifiers, operators and logical structures rather than individual characters.

The role of a tokeniser is to take the input text and break it up into tokens (words in natural language) so that the assembler or compiler using it only needs to concern itself with higher level structure and meaning. This division of labor is reflected in most programming language definitions in that they usually have a separate syntax definition for tokens and another for structures formed from the tokens.

The focus of this assignment is writing a tokeniser to recognise tokens that conform to a specific set of rules. The set of tokens may or may not correspond to a particular language because a tokeniser is a fairly generic tool. After completing this assignment we will assume that you know how to write a tokeniser and we will provide you a working tokeniser to use in each of the remaining programming assignments. This will permit you to take the later assignments much further than would be otherwise possible in the limited time available.

Writing Your Program

You are required to complete the implementation of the C++ files **tokeniser.cpp** and **tokeniser-basics.cpp** which are used to compile the programs **tokens** and **tokens-context**. You will complete the implementation of a function, ***next_token()***, that will read text character by character using the function ***nextch(),*** and return the next recognised token in the input. The tokens that must be recognised in the milestone and final submissions are specified in the file **includes/tokeniser.h**. Additional helper functions described in the **EBNF, Languages and Parsing (https://myuni.adelaide.edu.au/courses/54311/pages/ebnf-languages-and-parsing)** page are also provided as part of the precompiled library, their interfaces are shown in the **includes/tokeniser-extras.h** file.

The **tokeniser-basics.cpp** file is where you will implement the ***nextch()***, ***token_context()***, ***new_token()*** and ***initialise_tokeniser()*** functions. These are separated out so that it is possible to test your ***next_token()*** function without needing to complete all of the messy parts of these other functions.

Your **tokens** and **tokens-context** programs will be compiled using the **Makefile** in the zip file attached below using the command:

```
% make
```

Note: The only files you are allowed to edit are **tokeniser.cpp** and **tokeniser-basics.cpp**. All other files are automatically regenerated every time you run **make** and are not used by the **web submission system (https://cs.adelaide.edu.au/services/websubmission)** 's test scripts.

Testing Your Program

For each file in the **tests** directory, the output of the **tokens** and **tokens-context** programs must match the corresponding **.tokens** and **.context** output files respectively. You **must not** produce any output of your own. You can both compile and test your programs against all of the supplied tests using the command:

```
% make
```

The testing will not show you any program output, just whether or not a test was passed or failed. If you want to see the actual output, the commands used to run the tests are shown in string quotes ("). Simply copy the commands between the string quotes (") and paste them into your shell.

The web submission system will test your program in exactly this way. The key difference between your testing and the web submission testing is that the web submission system has some ***secret*** tests that it will use.

If you want to try additional tests, just create some new files in the **tests** sub-directory and generate the correct outputs using the command:

```
% make test-add
```

This will increase the number of tests that will be run in the future. You may add these new test inputs and outputs to **svn**.

Milestone Tokens

Your milestone submission will only be awarded marks for tests that require the correct recognition of the milestone tokens described in the **includes/tokeniser.h** file.

Note: the **includes/tokeniser.h** file describes

- the grammar rules for all tokens,
- the tokeniser interface functions that must be implemented,
- the rules for preprocessing special characters (not required for the milestone),
- the rules for differentiating identifiers and keywords (not required for the milestone),
- the rules for generating the context string for a token (not required for the milestone),
- the rules for modifying the spelling of a token (not required for the milestone) and
- the rules governing error handling.

Notes: all input must be read using the function ***next_ch()*** which must use the external function ***read_char()***.

Final Submission Tokens

Your final submission will be awarded marks for tests that require the correct recognition of all tokens and the correct implementation of the tokeniser interface functions described in the **includes/tokeniser.h** file.

Tests

In addition to the test files in the zip file(s) attached below, we will use a number of ***secret*** tests that may contain illegal characters or character combinations that may defeat your tokenisers. The ***secret*** tests may also check whether or not you have followed the rules for keyword recognition. **Note:** these tests are ***secret***, if your programs fail any of these ***secret*** tests you **will not** receive any feedback about these ***secret*** tests, even if you ask!

Startup Files

The newest of the following zip file(s) must be placed in the updates sub-directory and added to svn. When make is run, the newest zip file in the updates directory is used to update the startup files. Any files you are required to edit will not be updated but, a copy of the latest version of those files will be placed in the sub-directory originals.

- **assignment1-20200901-142352.zip (https://myuni.adelaide.edu.au/courses/54311/files/7362184/download?wrap=1)**
- assignment1-20200817-142446.zip