Assignment Schedule

All assignments, except where other instructions are given, are to be submitted electronically by 23:59 on the due date. Assignments handed in after the due date will be penalized **5% per day**.
Exception: late Assignment 2s will not be accepted after a solution has been posted.

You have to submit your assignments through MarkUs. Detailed instructions on how to submit each assignment will be provided later.

Assignments will be done by teams of 5 students. All students on a team will get the same mark for the assignment unless they all request a different division of marks in writing.

The teaching assistants will discuss testing of your compiler components and the format for handing in assignments. For every assignment you should hand in:
    a) a description of which member of your team was responsible for each part of the work.
    b) **well written** documentation for the work you did on the assignment.
Documentation will be worth at least 20% of the mark for every assignment. This percentage may be increased for some assignments.

Version Control

The use of some version control system (e.g. RCS, Subversion, Mercurial, git) is **MANDATORY** for this project. Use of a version control system is a really, really good idea for managing changing source code in a project of this scope. We will be totally unsympathetic about disasters arising from failure to use a version control system. Use of a version control system will be discussed in tutorial. MarkUs also provides a SVN repository that you can use for this purpose.

Project Testing

For each assignment in which you modify the project compiler, you must hand in test runs to demonstrate that you have implemented the features required for the particular assignment. As a minimum, these tests should illustrate:
    a) Correct handling of normal cases
    b) Correct handling of tricky or difficult cases
    c) Any error handling that you have implemented.
It is your responsibility to hand in sufficient information to convince the person marking the assignment that you have done a complete and correct solution to the assignment. You should always provide an index or manifest for the information you submit to make it easier for the teaching assistant to locate relevant information.

We expect each team to **thoroughly** test their software. The quality of each teams testing will be a significant component of the mark for each assignment.

The Assignments

Assignment 0, due January 12 [0%] Notify the Instuctor by email of the composition of your project team. Include the CDF user name for each team member. You need to form your group in MarkUs system which will be used for submitting the assignments. **It is important to form your team as soon as possible.** You can use the "Search for Teammates!" post in Piazza to find your team members.

Assignment 1, due January 18 [2%]. The purpose of this assignment is for you to gain familiarity with the project language. Write each of the following programs in the CSC488S Source Language. Store them electronically. These programs will be useful later for testing your compiler so you may want to think ahead, not only about syntax but about semantics as well. Try to use all of the features of the language in your programs. Small simple programs are better compiler test cases. These programs do not need to compute anything useful.

   a) a program that uses all arithmetic, logical and comparison operators

   b) a program using arrays including all forms of array declaration, positive and negative bounds.

   c) a program using all forms of loop building and loop exit constructs.

   d) a program using non-recursive functions and procedures with and without parameters

   e) a program using recursive functions and procedures with and without parameters
      include at least one nested procedure and one nested function declaration

The course project compiler uses the convention that source files in the project language end in the suffix `.488` e.g. `MyCleverProgram.488`

Assignment 2, due February 1 [6%] Using `jflex` and `jcup` build a parser for the language based on the definition provided. This parser should include the proper operator precedence and associativity so that it is LALR(1). Your parser can stop at the first syntax error, after reporting it (the default action). If the program is parsed successfully, your parser should not write anything. No semantic actions are required in the `jcup` file.

A set of files containing a skeleton compiler including an empty parser will be provided. Do NOT modify the lexical analyzer. Construct your parser using the skeleton provided.
A correct parser should not have any shift/reduce or reduce/reduce conflicts.
You MAY NOT use precedence and associativity features of `jcup` to resolve the precedence of arithmetic operators.
You may **not** change the programming language.

A solution to the assignment (i.e., a correctly working parser) will be made available on February 3.
Late Assignment 2s will not be accepted after the solution has been handed out.
To see an online manual on using `jcup` visit: http://www2.cs.tum.edu/projekte/cup/docs.php

Assignment 3, due February 22 [12%]. Documentation will be provided on the recommended structure of an **Abstract Syntax Tree** (AST) for the project language. and on the semantic analysis checking required for the project language.

A complete skeleton compiler including a working scanner and parser will be made available on February 3. Also included in this software will be skeleton software for building the AST Using this skeleton compiler you are to do the following:

  a) Modify the parser to generate a complete AST.

  b) Construct a semantics class (or classes) that implements semantic analysis for all language constructs. This class should not interface with the parser directly but should process the abstract syntax tree produced by the parser. NOTE that you will have to extend the AST data structures provided to do semantic analysis.

  c) Construct a symbol table class (or classes) that creates and manages the compilers symbol table

Add your classes to the skeleton compiler, and test them. **Note that this assignment is very thinking and programming intensive, so budget your time accordingly.**

Assignment 4, due March 15 [8%]. Documentation will be provided that describes the recommended code generation actions for the project language and the pseudo machine that is the target of code generation. Design code generation templates for the statements in the CSC488S Source Language. (A code generation template is a generic description of the machine code that will be generated for a language construct). Submit a written description of these templates including a description of any decisions you have made about how programs should be represented. Organize your code generation templates by source language constructs not by code generation action. The format of code generation templates will be discussed in the tutorial. The instructor will provide a small set of sample programs in the project language. Using your code templates show how each of these programs would be translated into the CSC488S Machine Language. Code generation templates will be discussed in detail in tutorial.

Assignment 5, due April 10 [12%]. Construct a code generator using your code template design from Assignment 4. Add it to your compiler and test it. You may **not** change the pseudo machine interpreter.

WARNING

Assignments 3, 4 and 5 are **very** design and programming intensive. Do **not** wait to start these assignments until just before they are due or you will **not** finish them. Thinking hard about code generation templates in Assignment 4 will make Assignment 5 much easier.

Notice: the instructor reserves the right to modify these assignments if *exceptional circumstances* occur.