Texas Christian University
Programming Language Concepts
CoSc 30403 – Spring 2017

Classtime: TR 11:00 AM - 12:20 PM
Classroom: TUC 245

Instructor: Dr. Comer
Office: TUC 208
Phone: 817-257-7166
Email: j.comer@tcu.edu

Office Hours: MW 10:30 AM - 12:00 Noon & 2:00 PM – 3:00 PM, TR 1:00 PM – 2:00 PM, and by appointment


Prerequisite: CoSc 20803 Data Structures

Grading:
Attendance at 1 CSS meeting 2%
Exam 1 18%
Exam 2 18%
Exam 3 18%
Term Project 20%
6-Programming assignments: 24%

Final Exam: Tuesday, May 9, 11:30AM – 2:00PM

Grades & Cell Phones
• Finals grades in this course will be the traditional letter grades. There will be no +/- distinctions.
• Cell phone usage is NOT allowed in class under any circumstances. TURN THEM OFF AND PUT THEM "OUT OF SIGHT" BEFORE ENTERING THE CLASSROOM!!

Prerequisites
CoSc 20803 - Data Structures (or equivalent!!)

What you need to already know - and what you will learn:

<table>
<thead>
<tr>
<th>Before attempting this course, students must be familiar with the following concepts</th>
<th>At the completion of this course, students should be able to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Write a program using pointers. - Use lists, stacks, queues, and tree structures in a program. - Write programs in at least two different high-level languages, at least one of which is block-structured.</td>
<td>- Demonstrate a knowledge of the factors that have influenced the development and evolution of programming languages and their application areas. - Describe appropriate problems and development environments for use with a</td>
</tr>
</tbody>
</table>
• Create and use user-defined data types.
• Use subprograms that employ at least two types of parameter passing (reference, name, value, or result).
• Compose functions mathematically.

• Write and trace code at an introductory level in a variety of programming languages.
• Discuss an article from the technical Computer Science literature.
• Describe the elements of abstract data typing and its advantages for software development.
• Discuss the application of Chomsky’s grammar classes to programming languages (emphasis on context-free and regular).
• Construct a simple lexical analyzer for a given language.
• Discuss the relationship between finite automata, regular languages, and state diagrams and their utility for scanner design.
• Describe language syntax using BNF/EBNF syntax diagrams, and transition graphs.
• Construct a parse tree for a legal statement in a language, given the grammar of the language.
• Construct a simple recursive descent parser for a given grammar.
• Trace the activation record stack and output for a block-structured program with nested subprogram calls and variable scopes.
• Be able to discuss and contrast the evolution of various programming language constructs (data types, assignment statements, control structures, subprogram implementation, etc).
• Be able to demonstrate an understanding of various parameter passing mechanisms (result, value/result, reference, name, etc.)

**Final Exam:** There are 3 equally weighted exams in the class.

**Comments:**

1. The course schedule is on the web and shows the topics, assignments, and due dates for all of the activities for this semester. Be aware that the schedule is tentative and subject to change as the instructor deems necessary - with appropriate notice.

2. There will be 3 equally weighted exams (weighted as indicated above). Material covered on one exam MAY be a requisite for other exam material - so if you didn't learn it for one exam it could affect your performance on the next.

3. **Programming assignments:** There will be 6 programming assignments in the class (the languages will depend on what compilers I am able to gain access to - most likely the majority of which will reside on the CoSc Department's computers in TUC 331). For the most part, the programming languages investigated in the class have been freely downloaded from various Internet sites (check the class website for URL's to some of these languages). Some are Linux implementations, while others are Windows. Macintosh users may find similar compilers available. *(You are free to use the machines in TUC 331 or to download and install the same software on your own individual machines.)*

The labs **MUST** be done as team efforts (2 persons per team - but with my approval) and will, for the
most part, be designed to afford you the opportunity to investigate programming language syntax and
semantics of languages that you may not have ever used. The purpose of the labs is to afford each of you
an opportunity to sample a variety of different languages!!

While labs will be done as team efforts, both members of the team are expected to make equal contributes
to the labs - otherwise, poor performance may result on the exams since you will be asked questions
concerning each of the assigned languages.

If you have NOT ever used Linux - I would encourage you to take this time to become familiar with it since
some of the assigned languages may only run under Linux. Use your time wisely!! Some of the computers
in TUC 331 are Linux systems while others are running Windows and Mac OS. All TUC 331 machines are
available to you for use during the semester.

Programming assignments are due at **class time on the date specified**. Late programs will NOT, under any
circumstances, be accepted. I REPEAT - LABS WILL NOT BE ACCEPTED LATE - DON'T ASK!!! All
source code and program output is to be submitted using **TURNIN** and must be submitted by **classtime** on
the due date.

In addition, at class time on the due date, you will be required to submit **hardcopy of your completed
assignment**. Your hardcopy printout must include:

- A copy of the input data file,
- A copy of the output produced by your program, and
- A copy of your commented program source code.

Note: you are expected to test and document all programming assignments consistent with what you have
done in other courses - **but with both members of the team clearly identified in the documentation!**
Comments are to be MEANINGFUL - not cute, but rather they should document what the program does.

**Undocumented** programming assignments will not be accepted and/or graded!!

You **MUST** document your programs consistent with the documentation model (check the COSC 10403
website for an example – or follow the link provide on this classes’ default page) used in other courses you
may have taken from me. **ASK ME IF YOU DON'T KNOW WHAT LEVEL OF DOCUMENTATION I
AM ASKING FOR!!!**

**Both team members are expected to contribute equal amounts of work on the programming assignment;**
as such, both should be identified in the program documentation, and both will share a common grade. **You**
will be required to write code, in each programming language, on exams – do not make the mistake of
failing to contribute on each assignment’s development.

4. **Team Presentation:** Each two-person team will be responsible for making a presentation of one of the
programming languages that have evolved over the years. BOTH MEMBERS OF THE TEAM WILL
BE EXPECTED TO PARTICIPATE IN THE PRESENTATION. The length of the presentation will
vary depending on the number of students enrolled in the class and the number of constituted teams.
Generally, presentations will range between 75-minutes (60 minute presentation + 15 minute Q&A) to 40-
minute (35 minute discussion + 5 minute Q&A).

More (or less) may be asked of a one or three-person team.

**NOTE:** Presentation length and programming languages to be studied will be defined after the class size
has been determined.

Presentations are to be professionally delivered (slides/transparencies/illustrations MUST be utilized) as
well as providing handouts (for everyone in the class) addressing syntax and semantic issues for the
assigned project language.
Proper, professional dress (NO SHORTS are permitted!), correct grammar (i.e., communication skills) will all play heavily into your team's grade. Since the classroom has resources for computer presentations (PowerPoint, Keynote, etc), you will be expected to use them. PEER ASSESSMENTS WILL BE EMPLOYED - they are to be taken seriously and teams should be graded honestly and fairly. Your assessments should not be out-of-line with my own!!

5. **Team Term Project:** A well-organized, well written (typed using a word processor) AND spellchecked report, double-spaced, and bound in an appropriate folder must be submitted on the prescribed due date. Your report MUST include a copy of all materials used in your classroom presentation (slides, handouts, etc). Your report should include example programs demonstrating your programming language's significant features and capabilities. In addition, an appropriate bibliography specifying all sources of materials used to obtain information for your presentation should be included. **Do NOT limit yourself to web references and Google - at a minimum 2 or 3 books (if available) MUST be referenced!** Grades will be determined by how well you describe the work and the overall organization of your report.

The course schedule includes milestones for when various sections of the report are due.

A "burned" CD containing everything pertaining to your project language is required on the date specified in the course schedule.

A list of reference papers that might be useful for your report are available (see the class website). While I am NOT asking you to read any of these reports, you may, nonetheless, find that one or more of these may provide good supplemental information for your report and presentation.

You will be provided a copy of the Term Paper Grading Criteria and Report Requirements that will be used for grading your presentation and report. Note, however, that you MUST include, at a minimum, 2 - 3 books (chosen from the TCU library) in your bibliography. If you have difficulties satisfying this requirement, please discuss this with me VERY early.

It is important to know that this project (and its report) is to be your own team's work. You may not plagiarize a presentation, or a document, intact directly from the web. To do so will result in a project grade of zero.

**All reports will be processed through "Turnitin" - a plagerism detection program that is commonly used by TCU faculty and the Koehler Writing Center and possibly the Stanford MOSS (Measure of Software Similarity) system.**

6. Plagiarism is taken very seriously by the Computer Science Department and by myself. You will only be given credit for work you do; handing in other people's work will result in an "F" for the assignment and possible disciplinary actions. The definitions and possible sanctions for plagiarism and other forms of academic misconduct are described in the section Academic Conduct Policy of the TCU Undergraduate Studies Bulletin. NEVER take credit for someone else's ideas or work as your own.

7. **Note:** Monday, April 10, 2017 is the last day to drop any class at TCU

8. The end of the semester is extremely hectic in all classes; plan accordingly.

**Disabilities Statement**

Texas Christian University complies with the Americans with Disabilities Act and Section 504 of the Rehabilitation Act of 1973 regarding students with disabilities. Eligible students seeking accommodations should contact the Coordinator of Student Disabilities Services in the Center for Academic Services located in Sadler Hall, 1010. Accommodations are not retroactive, therefore, students should contact the Coordinator as soon as possible in the term for which they are seeking accommodations. Further
information can be obtained from the Center for Academic Services, TCU Box 297710, Fort Worth, TX 76129, or at (817) 257-6567.

Adequate time must be allowed to arrange accommodations and accommodations are not retroactive; therefore, students should contact the Coordinator as soon as possible in the academic term for which they are seeking accommodations. Each eligible student is responsible for presenting relevant, verifiable, professional documentation and/or assessment reports to the Coordinator. Guidelines for documentation may be found at [http://www.acs.tcu.edu/disability_documentation.asp](http://www.acs.tcu.edu/disability_documentation.asp).

Students with emergency medical information or needing special arrangements in case a building must be evacuated should discuss this information with their instructor/professor as soon as possible.

**Note:** standard disclaimer that something might have to change.
You might be curious as to why I’ve selected the five programming languages (FORTRAN, Pascal, Ada, SNOBOL, and Bonus languages) required during the semester? Other languages have been used in the class: COBOL, PL/I, Algor, Prolog, LISP, Euphoria, Icon, Modula, Oberon, and Turing. However, it is difficult to acquire good compilers for many of these languages – not so, the ones mentioned below. All of the following languages have reasonable and readily available compilers.

**FORTRAN** - The FORTRAN team led by John W. Backus at IBM is generally credited as having introduced the *first complete compiler*, in 1957. *The list of high-tech tools in continuous use since the early 1950’s isn’t very long. Two tools on that list are the B-52 and FORTRAN.*

The language is a general-purpose, imperative programming language that is especially suited to numeric computation and scientific computing. Originally developed for scientific and engineering applications, The language came to dominate this area of programming early on and has been in continuous use for over half a century in computationally intensive areas such as numerical weather prediction, finite element analysis, computational fluid dynamics, computational physics and computational chemistry. *Contrary to popular thought – it is NOT a dead language.*

It is the preferred language by physicists and is widely used for large scale modeling of climate models, high performance computing, parallelized code programming, numerical simulation, etc.

**Pascal** - An influential imperative and procedural programming language, designed in 1968–69 and published in 1970 by Niklaus Wirth. The language is a small and efficient language intended to *encourage good programming practices* using structured programming and data structuring.

Initially, Pascal was largely, but not exclusively, intended to teach students structured programming. A generation of students used Pascal as an introductory language in undergraduate courses. Pascal was the primary high-level language used for development in the Apple Lisa, and in the early years of the Macintosh.

*While Pascal is rarely ever used now days, it is instructive to see the impact of Pascal on the evolution of other programming languages (Modula, Oberon Ada, C, etc.)*

**Ada** - A structured, statically typed, imperative, wide-spectrum, and object-oriented high-level computer programming language, extended from Pascal and other languages. It has built-in language support for strong typing, explicit concurrency, tasks, synchronous message passing, protected objects, and non-determinism. Ada improves code safety and maintainability by using the compiler to find errors in favor of runtime errors.

Its development was the result of the most extensive and expensive language design effort ever undertaken. Originally targeted at embedded and real-time systems. It was originally designed by a team led by Jean Ichbiah of CII Honeywell Bull under contract to the United States Department of Defense (DoD) from 1977 to 1983 to supersede the more than 450 programming languages then used by the DoD. Many did not have standards and software was rarely reused.

One of the first languages to include exception handling, generics, multi-tasking, and more. C++ borrows much from Ada. *Ada was named after Ada Lovelace (1815–1852), who is credited as being the first computer programmer. It is still in use for mission critical embedded systems (missile guidance, avionics, spacecraft navigation, missile interception, etc).*
SNOBOL - An unusual programming language that stands apart from most programming languages by having patterns as a first-class data type (i.e. a data type whose values can be manipulated in all ways permitted to any other data type in the programming language) and by providing operators for pattern concatenation and alternation. It was one of a number of text-string-oriented languages developed during the 1950s and 1960s.

The initial SNOBOL language was created as a tool to be used by its authors to work with the symbolic manipulation of polynomials. It had a simple syntax, only one datatype, the string, no functions, and no declarations and very little error control. The current version, SNOBOL4 has addressed many of the language’s early shortfalls.

*It was the first language to implement regular expressions (Chomsky Type 3 grammars) using patterns. The language is based on an overall logic derived from Markov Algorithms (a string rewriting system that uses grammar-like rules to operate on strings of symbols).*

*SNOBOL has influenced the development of other languages (i.e., SL5, Icon,*
Please sign on the line below and return this page to Dr. Comer. Signature signifies that you have thoroughly read and understand the above COSC 30403 Policies and Procedures.

You understand and agree with course policies and assignment due dates and deliverables.

______________________________________________________________
Student’s Name