Spring 2015 COP 3223 Section 4 Syllabus
Introduction to C Programming

Course Website: http://www.cs.ucf.edu/courses/cop3223/spr2015
Section Website: http://www.cs.ucf.edu/courses/cop3223/spr2015/section4

Course Prerequisites: No formal requirements except a background in precalculus mathematics

Lecturer: Arup Guha **Email:** dmarino@cs.ucf.edu
Office: HEC – 240 **Phone Number:** 321-663-7749 (cell)
Office Hours: http://www.cs.ucf.edu/~dmarino/ucf/OH.html

Note: I do NOT check my WebCourses email. Please email me at dmarino@cs.ucf.edu to contact me.

Course Description: COP 3223 provides an introduction to the C programming language for those with ***no prior programming experience***. The course aims to teach the syntax and use of major constructs of the C language and will not focus on algorithmic design (COP 3502 focuses on algorithmic design). Some of these constructs include: conditional statements, loops, functions, arrays, pointers, strings, structures, and file I/O.

Addendum to this section of the course: For students who already have programming experience, I will offer one opportunity for an "automatic A" in the course. During the first four weeks of the course, students who choose to attempt for this option will have to do one EXTRA program on top of those assigned to the whole class and take the final exam for course. If students perform well enough on these two components, then they will automatically be awarded an A for the course. If they do not, they will have already kept up with all of the regular course assignments and will simply be graded like the rest of the students in the course.

Recommended Course Textbook: Programming Knights: An Introduction to Computer Programming in Python and C

This custom textbook was specifically written for the course that we teach at UCF with the goal of bringing down cost and also providing a book that mirrored how the course is taught. This book will be cheaper than a new copy of the previous book, but potentially more expensive than a used copy of the previous book. Any of the previous textbooks for the course will be satisfactory for those who want a book of some sort. Here are two of these previous textbooks:

C Programming by K.N. King, ISBN 978-0-393-97950-3.
Problem Solving and Program Design in C by Hanly and Koffman, ISBN: 978-0-13-293649-1

Many students succeed by using freely posted materials online and making good use of Google searches. Whether or not you choose to get a book largely depends on how you prefer to learn.

Important Course Policies:

1) Cheating will not be tolerated. **If a student is caught cheating, then the grade on that assignment for all students knowingly involved (the person providing answers as well as the one taking the answers) will be a -25%.** (**Note, this is less than 0%.**) Since discussion of concepts with other students is often helpful, cheating must be more clearly defined. In particular, the following items are cheating: copying a segment of code of three lines or more from another student from a printout or by looking at their computer screen, taking a copy of another student's work and then editing that copy, and sitting side by side while writing code for assignments and working together on segments of code. In all of these situations, **BOTH people responsible**, the one from whom the three lines of code are taken as well as the person who takes those lines of code are engaging in academic misconduct. For example, if someone makes an electronic copy of their code accessible to ANYONE in the class (except for themselves) before an assignment is due, they are automatically culpable of academic misconduct. It does not matter if the recipient of the code doesn't use it, uses it a little, or copies it directly. Furthermore, based on the severity of the case, the entire course grade for the student may be lowered an entire letter grade. ***If you get stuck on an assignment, please ask either a TA or the instructor for help instead of getting help from another student.*** Part of the learning process in programming involves debugging on your own. In our experience, when a student helps another student with an assignment, they rarely allow the student getting help to "figure out" problems on their own. Ultimately, this results in a lack of debugging experience for the student receiving help. The goal of the TAs and instructors is to provide the facilitation necessary for students to debug and fix their own programs rather than simply solving their problems. **But, you are encouraged to work together on any non-graded programs to enhance the learning process.**

2) In order to take a make-up exam, you must request one from the instructor. Since this class is an on-line course, every attempt will be made to facilitate make-up exams, since it's possible that students' schedules won't allow them to come to campus at the appointed time for exams. **For each exam, directions for specifying make up exams will be posted in WebCourses. Please follow these directions to minimize scheduling errors.** If a student can't come to campus at all, they will be responsible for finding a remote proctor to administer their exam. **In the past, I offered alternate exams on the Wednesday, Thursday and Friday on campus before the Saturday exam. I will do the same this semester.**

3) Both the course web page and WebCourses will be crucial elements of the course. ***It is your responsibility to check both of these every three days.*** Some clarifications may only be announcements or videos via WebCourses, so make sure to check all announcements in a prompt manner.

Tentative Grading Criteria – Regular Section

Component	Percentage of Total Grade
Individual Programming Assignments (12)	36% (3% each)
Exam #1	20%
Exam #2	20%
Final Exam	24%

Programming Assignments

Each programming assignment is to be done individually. To make sure students are keeping up with the course, assignments will be due every Friday, except for the first Friday and the weeks with exams. **NO LATE ASSIGNMENTS WILL BE ACCEPTED!!!** Also, note that assignments are to be completed **individually.** If we fall behind, some due dates will be changed. To get the due date of each assignment, check WebCourses. **Due dates will NOT be posted anywhere else.**

Exams

For each exam, you'll be allowed to use a limited amount of notes as an aid. **No calculators will be allowed on any exam.** Other specifics for the allowable aids will be given online a few days before the exams. **Exams will be given ON CAMPUS on the following dates and times:**

Exam	Day	Date	Time	Location
Exam #1	Saturday	Feb. 14	8:00 – 9:15 am	**CB2-106**
Exam #2	Saturday	Apr. 4	8:00 – 9:15 am	**CB2-106**
Final Exam	Saturday	May 2	7:00 – 9:50 am	

Each exam will last for 75 minutes except for the final exam, which will last for two hours and fifty minutes.

If you can't make these times, some alternate dates/times will be provided. If you can't come to campus, you must arrange for a remote proctor. In these cases, you must provide for me the name and email address of your proctor BY January 31, 2015.

Letter Grades

I may assign +/- grades if I feel that they are appropriate. I assign letter grades a bit differently than other professors. I do not use a straight 90-100, 80-90, etc. grading scale. Rather, at the end of the semester, I chose my lines for each grade. The drawback to this technique is that students do not know exactly what letter grade they are earning during the semester. The advantage to it is that if I make a difficult exam, I can adjust my grades accordingly so students don't get punished for my exam making skills. In the past, my A line has ranged from about 83-88%, my B line has ranged from 67-72%, and my C line typically ranges from 52-55%. I do not guarantee that these will be accurate for this semester, but I wanted to give you a rough idea of how the grades have gone in the past. After each exam, I will update WebCourses to show you what letter grade you have at that point in time. If you have further questions about my grading philosophy, please read the document I have posted at http://www.cs.ucf.edu/~dmarino/ucf.

Tentative Grading Criteria – Experienced Section

Programming Assignments

Students need to submit each regular programming assignment by its due date as well as ONE EXTRA ASSIGNMENT (Charity Ball).

Exams

Experienced students will be given the final exam at **CB2-204** on February 7th from 8 am – 9:45 am.

Final Letter Grade

If students score at least an 80% on the final exam and 80% on the given programming assignment, they will automatically be awarded an A for the course and will not have to do any further assignments or exams. It may be the case that after giving the exam and seeing its difficulty I adjust the criteria for awarding automatic As. (For example, if I give a difficult final exam, I may lower the 80% threshold.)

If a student doesn't earn an A in this manner they will be required to complete the rest of the course and will be graded using the exact same criteria as the regular section of the course.

Tentative Schedule – Regular Section

Week	M/T	W/R/F	Saturday	Reading
Jan 12-26	Python/IDLE	Variables, I/O **Prog #1 Due**		Python – Cht 1
Jan 20-23	Formula Probs	If Statement **Prog #2 Due**		Python – Cht 2
Jan 26-30	If Statement	Loops **Prog #3 Due**		Python – Cht 3
Feb 2-6	Loops **Exp Prog Due**	Turtle **Prog #4 Due**	**Exp Final Exam**	Python – Cht 3
Feb 9-13	Intro to C	Vars, I/O etc. **Prog #5 Due**	**Exam #1**	C – Chts 1, 2
Feb 16-20	Arith Expr	if statement **Prog #6 Due**		
Feb 23-27	if statement	for loop **Prog #7 Due**		C – Cht 3
Mar 2-6	while loop	File I/O **Prog #8 Due**		C – Cht 4
Mar 9-13	**SPRING**	**BREAK!!!**		**NONE**
Mar 16-20	Arrays	Arrays		C – Cht 5
Mar 23-27	Functions **Prog #9 Due**	Functions		C – Cht 7
Mar 30-Apr 3	Functions **Withdrawal Deadline**	Functions	**Exam #2**	C – Chts 8, 9
Apr 6-10	Strings **Prog #10 Due**	More Strings		C – Cht 10
Apr 13-17	Arrays+Func	Strings+Func **Prog #11 Due**		C – Cht 11
Apr 20-24	Structs	Structs		C – Cht 12
Apr 27-May 2	Structs **Prog #12 Due**	**Thanksgiving**	**Final Exam**	

Note: These schedules are tentative.

Important Note for Beginning Programmers

Due to the high volume of students in the course and the limited resources we have for grading, the amount of graded work is minimal compared to what is actually necessary to learn how to program in C comfortably. We (the instructors and teaching assistants) strongly suggest that you write ***more*** programs than are actually assigned for you to turn in. The course web page will contain several suggestions of programs to write. Most lectures will contain sample programs that you can cut and paste, compile and edit. You may show the teaching assistants or course instructors extra programs you have written at any time. Since these aren't graded, the TAs can give you more feedback and help with them than on assignments. Also, it is encouraged that you write extra programs with others in the course; often times learning is facilitated by working with others. **You must work alone for all graded programming assignments, however.**

As the note above indicates, this class is far more time consuming than other introductory classes for many students. Unlike other introductory classes which simply ask students to read and regurgitate information, this class requires students to learn a new language, and then use that language to solve non-trivial problems that students have not seen before. It takes most students some time to get comfortable and confident with their programming skills. For each student, the amount of time and practice necessary to reach this level is different. Although we don't want to scare anyone, we do want to be up front about the fact that this class is a lot of work, and it's a different type of work than many other classes. (For example, it's possible you may get stuck on one error for a couple hours without making any progress. Once you fix that error, you might be able to finish your whole program in another thirty minutes.)

Important Note for Experienced Programmers

Although you may understand all the general concepts of programming taught in this class, if you've never programmed in C or Python, you may not know some of the very specific details that differentiate C and Python from other languages. We do write test questions that highlight these subtle differences and other subtleties of the C and Python languages. Make sure you read the textbook carefully so that you are aware of these details. We have had instances of a beginning programmer receiving the highest grade in the course, beating out many others who had been programming for three or four years because the beginner actually read the textbook and paid attention to those details that the experiences programmers missed.