Spring 2014 COP 3223 Section 4 Syllabus
Introduction to C Programming

Course Website: http://www.cs.ucf.edu/courses/cop3223/spr2014

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: TBA (see http://www.cs.ucf.edu/~dmarino/ucf by Friday Jan 10, 2014)

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: After some experimenting, I have adjusted the course so that I teach 5 weeks of the Python programming language in the beginning of the course, followed by 10 weeks of the C programming language. I believe the benefit of being exposed to two languages quickly outweighs the drawbacks of having fewer weeks of C. Last semester I had a lower retention rate than I had hoped. In part, I believe this is because I put together a rigorous syllabus, but rarely directly spoke with students to see how they were preparing for the course. This semester I am adding two required meetings/phone calls in hopes that I can improve the retention rate and the grades of the students who stay in the course. I believe that with these meetings, I can point to the type of extra work students will need to do to succeed.

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 48 hours after 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.

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.

Grading Criteria

Component	Percentage of Total Grade
Individual Programming Assignments (12)	24% (2% each)
Exam #1 Meeting/Phone Call	5%
Exam #2 Meeting/Phone Call	5%
Exam #1	10%
Exam #2	15%
Exam #3	15%
Final Exam	26%

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.**

Exam Meetings
In order to aid students, I will ask for each student to arrange for a meeting/phone call with me to discuss their progress after both the first and second exams. **If a student scores an 85% or higher on these respective exams, then they do NOT have to meet with me and will automatically be given this 5% credit.** Further details about arranging these meetings will be provided via WebCourses.

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	2/8/2014	8:00 – 9:15 am	ENG2 – 102
Exam #2	Saturday	3/15/2014	8:00 – 9:15 am	ENG2 – 102
Exam #3	Saturday	4/5/2014	8:00 – 9:15 am	ENG2 – 102
Final Exam	Saturday	4/26/2014	8:00 – 10:50 am	ENG2 – 102

Each exam will last for 75 minutes except for the final exam, which will last for three hours.

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 24, 2014.

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.

Final Criteria for Passing the Course

Finally, there is one more qualification for the grading criteria. **In order to pass the course you must earn at least a 40% on the final exam.** (Thus, if you earn a 75% in the class but only a 30% on the final exam, the final grade awarded will be a C- even though your percentage qualifies for a B. Also, if you earn a 50% on the final exam, but only a 30% for the course, you STILL do not pass the course. You would simply be awarded the letter grade that corresponded to the 30%, an F.)

Tentative Schedule

Week	M/T	W/R/F	Saturday	Reading
Jan 6-10	Python/IDLE	Variables, I/O		Python – Cht 1
Jan 13-17	Formula Probs	If Statement **Prog #1 Due**		Python – Cht 2
Jan 21-24	If Statement	Loops **Prog #2 Due**		Python – Cht 3
Jan 27-31	Turtle	Strings **Prog #3 Due**		Python – Cht 3
Feb 3 – 7	Dictionaries	Functions **Prog #4 Due**	**Exam #1**	Python – Cht 4
Feb 10-14	Intro to C	Vars, I/O etc.		C – Chts 1, 2
Feb 17-21	Arith Expr	if statement **Prog #5 Due**		C – Cht 3
Feb 24-28	for loop	while loop **Prog #6 Due**		C – Cht 4
Mar 3 – 7	**SPRING**	**BREAK!!!**		
Mar 10-14	File I/O	**Prog #7 Due**	**Exam #2**	C – Cht 5
Mar 17-21	Arrays **Withdrawal Deadline**	Arrays **Prog #8 Due**		C – Cht 7
Mar 24-28	2-D Arrays	Functions **Prog #9 Due**		C – Chts 8, 9
Mar 31-Apr 4	Functions	Functions **Prog #10 Due**	**Exam #3**	C – Cht 10
Apr 7-11	Strings	More Strings **Prog #11 Due**		C – Cht 11
Apr 14-18	Structs	Structs		C – Cht 12
Apr 21-26	**Prog #12 Due (4/21/2014)**		**Final Exam (9 am – 12 pm)**	

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, you may not know some of the very specific details that differentiate C from other languages. We do write test questions that highlight these subtle differences and other subtleties of the C language. 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.