JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Syllabus
Computer Science 601.220
Intermediate Programming
Fall, 2017
(4 credits, E)

Description

This course teaches intermediate to advanced programming, using C and C++. (Prior knowledge of these languages is not expected.) We will cover low-level programming techniques, as well as object-oriented class design, and the use of class libraries. Specific topics include pointers, dynamic memory allocation, polymorphism, overloading, inheritance, templates, collections, exceptions, and others as time permits. Students are expected to learn syntax and some language specific features independently. Course work involves significant programming projects in both languages. Recommended Course Background: EN.600.107 or EN.600.226, EN.600.111/EN.600.112 or equivalent.

Prerequisites

Introduction to Programming (AP CS, EN.600.107, EN.600.111/112, or equivalent).
Students are expected to be able to create and run simple programs in a general-purpose programming language (such as Java or Python) prior to enrolling in this course.

Instructors

Dr. Sara Miner More, Associate Teaching Professor
more@cs.jhu.edu, http://www.cs.jhu.edu/~more/, Office: Malone 221, 410-516-4248
Office hours: to be announced (check the course Piazza site)

Mr. Rohit Bhattacharya
rbhatta8@jhu.edu
Office hours: to be announced (check the course Piazza site)

Head Teaching Assistant

To be announced

Meetings

Section 1: Monday, Wednesday, Friday, 12:00–1:15pm, Maryland 310 (More)
Section 2: Monday, Wednesday, Friday, 1:30–2:45pm, Maryland 310 (More)
Section 4: Monday, Wednesday, Friday, 4:30–5:45pm, Maryland 310 (Bhattacharya)

Textbooks

- Recommended: Brian W. Kernighan & Dennis M. Ritchie, The C Programming Language, Prentice Hall, Inc., 2nd edition, 1988.
- Recommended: Andrew Koenig & Barbara E. Moo, Accelerated C++, Addison-Wesley, 2000. For a more detail-oriented presentation of C++, we recommend Deitel & Deitel, C++ How to Program; an electronic edition is available through MSEL.
- Additionally, you will be expected to read various materials posted on the course Piazza site.

Online Resources
Piazza (`piazza.com/jhu/fall2017/600220`) will be used for specific course material and course communications. Blackboard (`https://blackboard.jhu.edu`) will be used for assignment submission and grade reporting.

Course Goal
Students will have the ability to create large, complex, correct programs in C/C++.

Course Objectives Upon successful completion of this course, you should:

(1) Understand the C and C++ programming languages.
(2) Be able to read, write, trace, and debug C and C++ code.
(3) Be somewhat familiar with BASH and the Unix paradigm.
(4) Be familiar with a range of command-line tools for creating and debugging programs in C/C++.
(5) Be able to analyze an assignment specification and create a development plan for writing a program to implement it.
(6) Be able to create and use automated tests to verify the correctness of a program.
(7) Understand the importance of good programming practices such as modularity, testing, documentation, and incremental design.

ABET Outcomes

- An ability to apply knowledge of computing and mathematics appropriate to the discipline (a)
- An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution (b)
- An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs (c)
- An ability to function effectively on teams to accomplish a common goal (d)
- An ability to use current techniques, skills, and tools necessary for computing practice (i)
- An ability to apply design and development principles in the construction of software systems of varying complexity (k)

Course Topics

- General skills: code reading, tracing, and writing; pair programming and collaborative development; problem analysis and decomposition; incremental design; modular design; automatic testing
- Unix-like systems and tools: Unix basics, shell basics (command line tools, I/O redirection, etc.), shellscript and automation, text editors, gcc, make, gdb, valgrind, version control (git)
- C language, syntax, semantics, and concepts: types, operators, control structures, standard (terminal) I/O, scope, arrays, strings, pointers, dynamic memory allocation, structs, file I/O
- C++ language, syntax, semantics, and concepts: Standard Template Library (STL), containers (vs arrays), classes, templates, memory management, operator overloading, inheritance and polymorphism, object oriented design, exception handling

General Course Philosophy
This course will focus more on learning than on assessment. While we will use grades to let you know how you are doing, we hope that your goal is learning the material, rather than "getting a good grade." In the end, the best way to get a good grade is to develop an interest in learning and engage with the material in a self-directed fashion. Besides, in the long run the knowledge and skills you acquire are far more important than the grade is.

That said, be aware that the main difficulty many students have with this course is time management. It will be a lot of work, and if you don't budget your time well, you may find yourself with a grade that

does not reflect how well you understand the material. Additionally, while there are lots of resources provided to help you succeed, we cannot force you to use them; it is important to avail yourself of these resources (particularly office hours).

Expectations
Students will be expected to complete a variety of computer programming assignments, as well as written homeworks. Some assignments may be done in pairs or groups, others must be completed alone. See the specific assignment page for details of what is permitted for a particular assignment. Failure to follow these guidelines may be a violation of the academic ethics code.

While code reuse is an important feature of modern programming practices, for this course you will be expected to write most of the code for your homework assignments from scratch. You may use language libraries (according to assignment specifications), and you may always re-use your own code from prior work in the course. Downloading partial solutions from the internet, however, is an ethics violation. Using code from in-class examples or from the textbook is acceptable, but you must cite the source properly (in a comment in your code describing the original source).

There will be two midterm exams. We will be doing lab-work during class, as well as lecture and discussion. Students are expected to attend all class sessions, and participate in exercises and activities (though we understand the occasional absence due to illness). If you must miss more than one class in a row, please contact the instructor. There will be no final exam in this course; however, during the finals period, you and your final project team will be expected to meet with your instructor to formally review your final project submission. Final project code review appointments will be distributed throughout the finals week, based on availability of students and staff.

Homework assignments are expected to take at least 10 hours per week; some students report taking significantly longer, so start early, and budget your time well. Additionally, try to use incremental development so that even if you run out of time, you can still turn in code that implements some of the desired functionality (with a README and comments explaining what's missing). Half the features working all the way gets you a lot more partial credit than all the features half-way working.

Students are expected to learn material outside of class time and homework, as well. We will generally provide links to tutorials, references, and other resources for each topic, as well as listing relevant textbook chapters in the schedule. Students are expected to read these, as well as seek out other resources on their own to further their understanding of topics. There is a wealth of programming information on the internet; if one explanation doesn't make sense, you can probably find another that does.

Homework policy
Assignments will be due by 11:00pm on the due date (unless otherwise indicated on the assignment page). There will be a 59 minute grace period after the deadline during which assignments can still be submitted, but will receive a 10% penalty. After that, submissions will not be accepted.

Given this policy, please plan to get your homework done and turned in early, so that if you encounter any last-minute delays, it will not hurt you too badly. Additionally, Blackboard will allow multiple submission attempts; we will simply grade the last one. So it's a good idea to develop your program incrementally, and turn in a working (if only partially complete) version every few days.

Exceptions can only be made by an instructor (not TAs/CAs), and will only be considered in circumstances outside the control of the student (e.g. illness, death of a relative, etc.). No exceptions will be given for failure to plan ahead, or simply having "too much work." If you must request an exception,

do so as early as possible; it is easier to get an exception if you ask before an assignment is due, rather than after.

Detailed information about homework policies is given on the Course Homework Policies page, which is linked from Piazza.

Grading

- 5% - participation during class
- 35% - homework (due dates vary; will be listed on Piazza)
- 15% - midterm 1 (during class, Wednesday, October 18)
- 12% midterm project (in teams)
- 15% - midterm 2 (during class, Friday, December 1)
- 18% - final project (in teams)

All scores and grader commentary on your homework submissions will be available via Blackboard. Please keep your own record of your grades so that you will know your standing in the course.

Ethics

The strength of the university depends on academic and personal integrity. In this course, you must be honest and truthful, abiding by the *Computer Science Academic Integrity Policy*:

> Cheating is wrong. Cheating hurts our community by undermining academic integrity, creating mistrust, and fostering unfair competition. The university will punish cheaters with failure on an assignment, failure in a course, permanent transcript notation, suspension, and/or expulsion. Offenses may be reported to medical, law or other professional or graduate schools when a cheater applies.
>
> Violations can include cheating on exams, plagiarism, reuse of assignments without permission, improper use of the Internet and electronic devices, unauthorized collaboration, alteration of graded assignments, forgery and falsification, lying, facilitating academic dishonesty, and unfair competition. Ignorance of these rules is not an excuse.
>
> Academic honesty is required in all work you submit to be graded. Except where the instructor specifies group work, you must solve all homework and programming assignments without the help of others. For example, you must not look at anyone else's solutions (including program code) to your homework problems. However, you may discuss assignment specifications (not solutions) with others to be sure you understand what is required by the assignment.
>
> If your instructor permits using fragments of source code from outside sources, such as your textbook or on-line resources, you must properly cite the source. Not citing it constitutes plagiarism. Similarly, your group projects must list everyone who participated.
>
> Falsifying program output or results is prohibited.
>
> Your instructor is free to override parts of this policy for particular assignments. To protect yourself: (1) Ask the instructor if you are not sure what is permissible. (2) Seek help from the instructor, TA or CAs, as you are always encouraged to do, rather than from other students. (3) Cite any questionable sources of help you may have received.
>
> On every exam, you will sign the following pledge: "I agree to complete this exam without unauthorized assistance from any person, materials or device. [Signed and dated]". Your course instructors will let you know where to find copies of old exams, if they are available.

In addition, the specific ethics guidelines for this course are:

(1) This course is about learning, and encourages collaboration toward that end. In general, if a collaborative act helps you to learn, it is probably permitted. If, on the other hand, it helps

you avoid learning, it is not permitted. For example, helping your friend learn how to use the debugger is great. Helping your friend by debugging their code for them is bad, because your friend will never learn how to do it by watching you. A main focus of this course is learning skills, and you can't acquire skills without practice. Therefore, "helping" other students by allowing them to skip the practice endangers the learning outcomes of the course, and is prohibited. Helping other students practice more efficiently and effectively (e.g. not waste 3 hours trying to fix one bug), on the other hand, actively supports the learning goals of the course, and is not only allowed, but encouraged. Please remember to put a "thank you" to everyone who helped you with an assignment in the README for that assignment.

In general, when helping another student, never do something for them; instead, try to think like a teacher and "teach" them how to do it themselves. This will help you both learn, since teaching something is a great way of learning more about it as well.

- Many of the assignments for this course will allow for and encourage collaboration.
- Each assignment will specify how many students may work collaboratively on a single submission.
- Students working together on a submission are considered a "team," and there are no limits to communication, cooperation, and sharing within a team.
- Even within a team, you should try to help each other learn, not do work on each other's behalf.
- Therefore, work should be done "as a team," with all members present; don't just split the assignment in half and each do half of it.
- Outside of your team, you are still encouraged to discuss class material, including assignments, algorithms, techniques, and tools.
- Please do not share working source code outside your team, however.
- Asking a friend to let you look at their working code is not allowed, nor is offering to let someone else look at your working code.
- Asking a friend to help you debug your non-working code is fine, but it should work by you "driving" and having your friend help by "navigating."
- In general, when helping others, think "teach, not "do".
- Always thank anyone who helped you on a given assignment in your README file.

Report any violations you witness to the instructor.
You can find more information about university misconduct policies on the web at these sites:

- Undergraduates: `e-catalog.jhu.edu/undergrad-students/student-life-policies/`
- Graduate students: `e-catalog.jhu.edu/grad-students/graduate-specific-policies/`

Students with Disabilities
Any student with a disability who may need accommodations in this class must obtain an accommodation letter from Student Disability Services, 385 Garland, (410) 516–4720, `studentdisabilityservices@jhu.edu`.