JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Syllabus
Computer Science EN.601.220
Intermediate Programming - Online
Fall, 2020 (4 credits, E)

*(*The instructor reserves the right to make adjustments to this syllabus as deemed necessary with notice.)

Instructors

Dr. Ali Darvish, Lecturer
darvish@jhu.edu, www.cs.jhu.edu/~darvish/,
Office hours: to be announced (check the course Website)

Sing Chun Lee, Ph.D. Candidate
singchun.lee@jhu.edu, www.cs.jhu.edu/~singchun,
Office hours: to be announced (check the course Website)

Head Course Assistants

Kaushik Srinivasan (kriniv4@jhu.edu)
Office hours: to be announced (check the course Website)

Raghav Sambasivan (rsambas1@jhu.edu)
Office hours: to be announced (check the course Website)

Meetings

Online meetings on Zoom.

Section 1: Monday, Wednesday, Friday, 12:00–01:15pm, Zoom (Darvish)
Section 2: Monday, Wednesday, Friday, 01:30–02:45pm, Zoom (Darvish)
Section 4: Monday, Wednesday, Friday, 10:00–11:45am, Zoom (Lee)

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: Link.
- Additionally, you will be expected to read various materials posted on the course website.

Online Resources

The course website is jhu-ip.github.io/cs220_f20/. Piazza (piazza.com/jhu/fall2020/601220) will serve as our primary communication channel. You should sign up for Piazza immediately. Gradescope (gradescope.com) will be used for assignment submission; you will receive an email invitation to this course on Gradescope.

Course Information

- This course teaches intermediate to advanced programming, using C and C++. (Prior knowledge of these languages is not expected, but prior programming experience and a general understanding of basic object-oriented programming are 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 the syntax and some language-specific features independently. Course work involves significant programming projects in both languages.
- **Prerequisites**
  Introduction to Programming (AP CS, EN.600.107/EN.601.107, 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.
- **Required, Elective or Selective Elective**
  EN.600.107/EN.601.107 or EN.600.111/EN.600.112 (Introduction to Programming) and/or
  EN.600.226/EN.601.226 (Data Structures) and/or
  equivalent

Course Goals

Specific Outcomes for this course are:

- Students will learn to read, write, trace, test, and debug C and C++ program codes.
- Students will learn to use command-line Unix/Linux tools for file management, creating and debugging programs in C/C++.
- Students will learn to analyze a programming specification, design a program that fulfills the specification, create a development plan, execute the plan, implement and deliver the program on time.
- Students will learn to utilize a distributed version control system to manage the development of a program.
- Students will learn to understand the importance of good programming practices such as modularity, testing, documentation, and incremental design.

This course will address the following ABET Outcomes:

(1) (SQ1) Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.
(2) (SQ2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. in computing practice based on legal and ethical principles.
(3) (SQ5) Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline.
(4) (SQ6) Apply computer science theory and software development fundamentals to produce computing-based solutions.

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.

Course Expectations & Grading

All students are generally expected to attend all the online sessions of this course and actively participate by answering and asking questions and engaging with material via the posted material, exercises, assignments, and projects. Our online meeting time may involve any combination of recap of the material, discussion on the important concepts, review of previous exercise answers and/or assignments, as well as live coding exercises. Before each online session, students are expected to go through the related material posted on the course website and/or Piazza and finish the coding exercises.

Students will be expected to complete a variety of computer programming assignments, as well as written homework assignments. 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, 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 reuse your own code from prior work in the course. Downloading full or partial solutions from the internet, however, is an ethics violation. Using code from class examples, slides, or the textbook is acceptable, but you must cite the source properly (in a comment in your code describing the original source).

There will be one midterm exam and a final exam. Midterm exam will cover the topics of the first half of the class (C material) and the final exam is a comprehensive exam which will cover all the course material discussed in the class. The exams are desigend to assess your knowledge of programming in C and C++ as well as your problem solving ability, not your knowledge of specific tools or technologies discussed in class.

Homework assignments are expected to take a considerable amount of time; some students report taking as much as 40 hours or more per week, so start early and budget your time well. On average, it should

take at least 10 hours per week. 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 file and comments explaining what's missing). Keep in mind that half the features working all the way get you a lot more partial credit than all the features half-way working. Good use of the version control system will significantly help with your incremental development. For each commit, you should maintain your README file to outline what's missing and what's done, and try to keep each commit a submittable version. This is a good practice, and can come in very handy in case you accidentally delete the local copy of your homework!

Students are expected to learn material outside of online sessions time and homework, as well. We will generally provide links to tutorials, references, and other resources for each topic. 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 11:00 pm on the due date** (unless otherwise indicated on the course Piazza site). Non-compiling code may earn a score of zero, so students are strongly encouraged to double-check that all submitted codes fully compile with no errors or warnings in the standard course compilation environment. There will be a 30-minute grace period after the deadline during which assignments can still be submitted, but will receive a penalty of a deduction of 10% of the points possible on the assignment. In addition, each student will be permitted to use up to 3 "late days" total during the semester on the individual coding assignments only. (That is, no late days are allowed for handwritten homework or partner-based coding projects.) Each delay of up to 24 hours past the 11:00 pm deadline for an individual coding assignment counts as one late day. Students may not use a grace period with a late day. For example, if Student A turns in her homework at 11:45 pm on the day it is due (only 45 minutes past the original deadline), then she will be using one late day. Furthermore, if Student B turns in his homework at 11:15 pm on the day after it is due, he'll be using two late days.

Given these policies, 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, Gradescope 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 fully-compiling (if only partially complete) version every day or so.

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

In-class Exercises

Many of the course topics will be supported by an exercise. Although these exercises do not count towards your course grade, they are a very important part of your learning, and as such we *strongly* recommend that you complete them fully. We will review the solutions to these exercises either fully or just the important parts in the online sessions. In general, we will *not* post solutions to the in-class exercises — if you are having trouble completing an exercise, seek help office hours, or on Piazza.

Grading Breakdown

- 9% - handwritten homework (due dates vary; will be listed on Piazza)
- 31% - coding homework (due dates vary; will be listed on Piazza)
- 13% - midterm coding project (in teams)
- 15% - midterm exam on TBA
- 17% - final coding project (in teams)
- 15% - final exam on TBA (must be passed to pass the course; passing threshold is 70%)

All scores and grader commentary on your homework and project submissions, as well as exams, will be available via Gradescope. Please keep your own record of your grades so that you will know your standing in the course. At the end of the term, letter grades are generally assigned according to the following scale. You should not expect a curve in this course.

$[97, 100]$: A+,	$[93, 97)$: A,	$[90, 93)$: A-
$[87, 90)$: B+,	$[83, 87)$: B,	$[80, 83)$: B-
$[77, 80)$: C+,	$[73, 77)$: C,	$[70, 73)$: C-
$[67, 70)$: D+,	$[60, 67)$: D,	$[0, 60)$: F

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 as follows:

(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.

(2) 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.
   - 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.
   - The two coding projects for this course will allow for and encourage collaboration in teams. There are no limits to communication within a team, but work should be done "as a team" with all members present; don't just split the assignment into pieces and work on separate parts. No working source code should be shared outside your team.

Report any violations you witness to the instructor.

You can find more information about university misconduct policies on the web at these sites:

- For undergraduates:
  http://e-catalog.jhu.edu/undergrad-students/student-life-policies/
- For graduate students:
  http://e-catalog.jhu.edu/grad-students/graduate-specific-policies/

Personal Wellbeing

- If you are sick, in particular with an illness that may be contagious, notify me by email but do not come to class. Rather, visit the Health and Wellness: 1 East 31 Street, 410-516-8270. See also http://studentaffairs.jhu.edu/student-life/support-and-assistance/absences-from-class/illness-note-policy/
- All students with disabilities who require accommodations for this course should contact me at their earliest convenience to discuss their specific needs. If you have a documented disability, you must be registered with the JHU Office for Student Disability Services (Shaffer 101; 410-516-4720; http://web.jhu.edu/disabilities/) to receive accommodations.
- If you are struggling with anxiety, stress, depression or other mental health related concerns, please consider visiting the JHU Counseling Center. If you are concerned about a friend, please encourage that person to seek out our services. The Counseling Center is located at 3003 North Charles Street in Suite S-200 and can be reached at 410-516-8278 and online at http://studentaffairs.jhu.edu/counselingcenter/

Classroom Climate

I am committed to creating a classroom environment that values the diversity of experiences and perspectives that all students bring. Everyone here has the right to be treated with dignity and respect. I believe fostering an inclusive climate is important because research and my experience show that students who interact with peers who are different from themselves learn new things and experience tangible educational outcomes. Please join me in creating a welcoming and vibrant classroom climate. Note that you should expect to be challenged intellectually by me, the TAs, and your peers, and at times this may feel uncomfortable. Indeed, it can be helpful to be pushed sometimes in order to learn and grow. But at no time in this learning process should someone be singled out or treated unequally on the basis of any seen or unseen part of their identity.

If you ever have concerns in this course about harassment, discrimination, or any unequal treatment, or if you seek accommodations or resources, I invite you to share directly with me or the TAs. I promise that we will take your communication seriously and to seek mutually acceptable resolutions and accommodations. Reporting will never impact your course grade. You may also share concerns with the Department Head (Randal Burns, randal@cs.jhu.edu), the Director of Undergraduate Studies (Joanne Selinski, joanne@cs.jhu.edu), the Assistant Dean for Diversity and Inclusion (Darlene Saporu, dsaporu@jhu.edu), or the Office of Institutional Equity (oie@jhu.edu). In handling reports, people will protect your privacy as much as possible, but faculty and staff are required to officially report information for some cases (e.g. sexual harassment).