CSCI 261: Programming Concepts
Fall 2019 Syllabus
https://cs-courses.mines.edu/csci261/index.html

This course introduces fundamental computer programming concepts using a high-level language and a modern development environment. Programming skills include sequential, selection, and repetition control structures, functions, input and output, primitive data types, basic data structures including arrays and pointers, objects, and classes. Software engineering skills include problem solving, program design, and debugging practices.

Well, that's the official, *boring* description. The goal of this course is to open your mind to *computational thinking*, to educate you to leverage programs as tools in your field of study, and to empower you with a fundamental knowledge of programming.

Course Learning Outcomes
By the end of this course, students will be able to:

- Identify and construct proper object-oriented C++ syntax. Explain the components that comprise C++ syntax and how the components operate together.
- Design and write pseudocode to accomplish a given task or solve a defined problem using common programming design structures including conditionals, loops, functions, arrays, and classes.
- Translate pseudocode into valid and correct C++.
- Analyze & trace existing code and calculate the output given an initial input while explaining what the code does.
- Identify and correct errors in C++ syntax, program structure, and algorithm.
- Discuss at a high level how C++ code becomes an executable program and how data is stored in computer memory.

Student Evaluation
The final course grade will be computed from the following course percentage breakdown:

- 5% zyBook Completion
- 25% Assignments + Labs
- 10% Final project
-
- 10% Quizzes
- 15% Exam I
- 15% Exam II
- 20% Final Exam

Final grades will be determined using a straight scale. The straight scale assigns letter grades as follows:

- [93, 100] -- 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+
- [63, 67) -- D
- [60, 63) -- D-
- [0, 60) -- F

<u>You must pass both the final exam and the final project with at least a 60% grade on each in order to pass this course.</u>

All assignments are due at 11:59PM on the date stated. In order to receive full credit for any assignment, the submission must be on time, unless an approved absence is submitted. Submissions will be accepted for an additional 72 hours subject to the following late penalties:

- **(00h 00m, 24h 00m) Late: -10%**
- **[24h 00m, 48h 00m) Late: -25%**
- **[48h 00m, 72h 00m) Late: -50%**
- **[72h 00m, INF) Late: -100%**

Students are given a set of 5 Skip Days. Each Skip Day used allows a submission to be submitted an extra day without the late penalty being applied.

While there exist many compilers and IDEs, it is possible your code and solution may work in one environment but not another. **All submissions will be graded against g++ (the compiler used with CLion)**. It is your responsibility to ensure your submission works in the lab environment. If your submission does not work with g++, the following penalties will be applied and the grader will contact you to correct your submission:

- **Submitting Extraneous Files: -5%**
- **Missing files: -10%**
- **Compiler Error: -25%**

For a discrepancy in any grade in which you think you deserve more points than you received, **you must raise the issue within one week of the day the item was returned**. No claims, justifiable or not, will be considered after this deadline. For discrepancies with assignments, you should contact the grader first. For any other discrepancies, you should contact your instructor.

Collaboration Policy for Programming Assignments & Projects in CS Courses

The following policy exists for all CS courses. This policy is a minimum standard; your instructor may decide to augment this policy.

- If the project is an individual effort project, you are not allowed to give code you have developed to another student or use code provided by another student. If the project is a group project, you are only allowed to share code with your group members.
- You are encouraged to discuss programming projects with other students in the class, as long as the following rules are followed:
  - You view another student's code only for the purpose of offering/receiving debugging assistance. Students can only give advice on what problems to look for; they cannot debug your code for you. **All changes to your code must be made by you. A meeting with another student regarding an assignment must follow the "empty hands" policy where you walk away with no artifact of the discussion.**
  - Your discussion is subject to the empty hands policy, which means you leave the discussion without any record [electronic, mechanical, or otherwise] of the discussion.
- Any material from any outside source such as books, projects, and in particular, from the Web, should be properly referenced and should only be used if specifically allowed for the assignment.
- To prevent unintended sharing, any code stored in a hosted repository (e.g. on github) must be private. For group projects, your team members may, of course, be collaborators.

- If you are aware of students violating this policy, you are encouraged to inform the professor of the course. Violating this policy will be treated as an academic misconduct for all students involved. See the Student Handbook for details on academic dishonesty.

Academic Code of Honor

- All students are expected to follow the University's Academic Code of Honor.
- A student or assigned team working on a program may discuss high-level ideas with other students or teams. However, at time of submission all work submitted **must be his/her/their own work**.
- Use of the Internet as a reference is allowed but directly copying code or other information is **cheating**. It is cheating to copy, allow another person to copy, all or part of an exam or a project, or to fake program output. It is also a violation of the Code of Honor to observe and then fail to report academic dishonesty. *You* are responsible for the security of your own work.
- We will provide, as part of the course, functional code examples for most of the topics covered. While you are encouraged to examine these examples, your submissions must represent a good-faith effort to complete the assignment. Merely copying and pasting code from the examples will result in a failing grade. Furthermore, relying too heavily on the given examples will fail to prepare you for the much more open-ended final project.
- Developing a program is a creative exercise; just like in art, no two programs will look exactly the same (unless the "canvas" has been copied). To ensure copying does not exist, homework assignments are checked via an automated system that generates similarity metrics between your work and that of all other students and previous student work in this class. When a high-level of similarity is detected, the course coordinator is notified and investigates the similarity. If plagiarism is evident, the course coordinator begins the process of submitting an Academic Misconduct.

Disclaimer

This syllabus is intended to give the student guidance in what may be covered during the semester and will be followed as closely as possible. However, the professor reserves the right to modify, supplement and make changes as the course needs arise.