Cheating Quiz

Gabriel Hugh Elkaim
• The working definition for cheating is presenting someone else's work as your own.

• Here are a few scenarios, for each of these let me know if you think this is cheating (per definition above) or not:

• For each Scenario please raise your hand if you think it is cheating.
UCSC Academic Misconduct Penalty Levels

1. Warning letter and other potential ramifications
2. Mandatory Suspension
3. Permanent Dismissal from the University of California.

- Typically the 1st time is a level one. The next infraction will be a level two. A grievous 1st time case can also be counted as a level two directly.
Scenario 1

- You have a friend in your sorority/fraternity who took the class two years ago, and they passed on the solutions they implemented for the labs. You do the lab with their code open in front of you, and change details around as appropriate, but essentially copy their code.
Scenario 1

- You have a friend in your sorority/fraternity who took the class two years ago, and they passed on the solutions they implemented for the labs. You do the lab with their code open in front of you, and change details around as appropriate, but essentially copy their code.

- This is an academic violation because the code is theirs not yours. It does not represent your original work.
Scenario 2

- You have a friend in your sorority/fraternity who took the class two years ago, and they passed on the solutions they implemented for the labs. Before starting your lab assignment, you look through their solution to get inspiration and ideas on how to implement the solution.
Scenario 2

- You have a friend in your sorority/fraternity who took the class two years ago, and they passed on the solutions they implemented for the labs. Before starting your lab assignment, you look through their solution to get inspiration and ideas how to implement the solution.

- This is not as obvious but still an academic violation. Again the code base you are using is not your own.
Scenario 3

- You and a friend decide to work together on the assignment. You code together, and submit the same files.
Scenario 3

- You and a friend decide to work together on the assignment. You code together, and submit the same files.
- This is also an academic violation. The work you submitted is not your own.
Scenario 3(a)

• You and a friend decide to work together on the assignment. You code together, and submit the same files (except your files note your collaboration with your friend).
Scenario 3(a)

- You and a friend decide to work together on the assignment. You code together, and submit the same files (except your files note your collaboration with your friend).
- This is not an academic violation. You will receive a zero this assignment if the assignment does not allow collaboration.
Scenario 4

• You and a friend decide to work together on the assignment. You discuss how to approach the different sections, what strategies you might use, and help each other clarify what the assignment demands. You go off and code on your own, and submit your individual files.
Scenario 4

- You and a friend decide to work together on the assignment. You discuss how to approach the different sections, what strategies you might use, and help each other clarify what the assignment demands. You go off and code on your own, and submit your individual files.

- This is perfectly fine. You are not discussing specific details but what is required of the assignment.
Scenario 4(a)

- You and a friend decide to work together on the assignment. You discuss how to approach the different sections, what strategies you might use, and help each other clarify what the assignment demands. You go off and code on your own, and submit your individual files (noting in the README that you discussed the assignment with your friend).
Scenario 4(a)

- You and a friend decide to work together on the assignment. You discuss how to approach the different sections, what strategies you might use, and help each other clarify what the assignment demands. You go off and code on your own, and submit your individual files (noting in the README that you discussed the assignment with your friend).

- This is even better. If there are similarities in your files we know who you worked with.
Scenario 5

- You and a friend decide to work together on the assignment. You discuss how to approach the different sections, what strategies you might use, and help each other clarify what the assignment demands. You write your code sitting next to each other and checking in to make sure you are not making mistakes.
Scenario 5

- You and a friend decide to work together on the assignment. You discuss how to approach the different sections, what strategies you might use, and help each other clarify what the assignment demands. You write your code sitting next to each other and checking in to make sure you are not making mistakes.
- This is an academic violation. You are sharing code with each other and the work is no longer your own. Both of you fail.
- (noting in the README that you programmed together will get you a zero on the assignment, but not an academic violation).
Scenario 6

- You are struggling with the assignment, and ask one of your friends for help. They show you the code they wrote for the section you are having difficulties with, and you see how to do it. Just to help you out. You go off and code up your solution, and turn it in.
Scenario 6

- You are struggling with the assignment, and ask one of your friends for help. They show you the code they wrote for the section you are having difficulties with, and you see how to do it. Just to help you out. You go off and code up your solution, and turn it in.

- This is an academic violation. This is again not your work. Not only will you fail but the person who gave you the code will fail as well.
Scenario 7

• You are struggling with the assignment, hire someone on the internet to do the programming assignment for you. After getting back the program, you study it carefully and make sure you understand what the code does before turning it in.
Scenario 7

• You are struggling with the assignment, hire someone one the internet to do the programming assignment for you. After getting back the program, you study it carefully and make sure you understand what the code does before turning it in.

• This is an academic violation. The work you turned in is not your own (and expect to be blackmailed by the programmer).
Scenario 8

- You are struggling with the assignment, and search the web for solutions. You find a solution or something rather close to it on StackExchange. You use this code but make some small alterations to it.
Scenario 8

- You are struggling with the assignment, and search the web for solutions. You find a solution or something rather close to it on StackExchange. You use this code but make some small alterations to it.

- This is an academic violation. The work is not your own. Further, it is likely that someone else will also find the same work and you will both get flagged for it. Don’t go looking on the internet for solutions.
Scenario 9

- You are struggling with the assignment, and go to one of the “cheating” sites like CourseHero which has solutions to class assignments and find a solution to the lab.
Scenario 9

- You are struggling with the assignment, and go to one of the “cheating” sites like CourseHero which has solutions to class assignments and find a solution to the lab.

- This is an academic violation. The work is not your own. CourseHero is pure evil, don’t even think about looking at it.
Scenario 10

- You are struggling with the assignment, and ask your new girlfriend to help you out (she didn’t take the class but knows how to code). She sits next to you and tells you how to code it up (never touching the keyboard).
Scenario 10

- You are struggling with the assignment, and ask your new girlfriend to help you out (she didn’t take the class but knows how to code). She sits next to you and tells you how to code it up (never touching the keyboard).

- This is an academic violation. The code is not your own. If the TA’s see this happening, they will immediately flag you for cheating and banish your girlfriend from the lab.
CMPE012/L

Computing Systems and Assembly Language Programming

Gabriel Hugh Elkaim
About: Gabriel Elkaim

- Undergraduate from Princeton University in Mechanical and Aerospace Engineering (MAE) - 1990


- Developed an autonomous wing-sailed catamaran for PhD thesis
The Course

• Introduction to computer systems and assembly language and how computers compute in hardware and software.

• Topics include digital logic, number systems, data structures, compiling/assembly process, basics of system software, and computer architecture.

• Will include a very basic introduction to C (mostly to map from assembly language to a higher level language).

• Prerequisite(s): previous programming experience; concurrent enrollment in course 12L required.
You did not start

Commit early and often
Waitlist

- Lab classes manage waitlist from the lab (not lecture)
- Single waitlist covers all section of lab
- When you come off the waitlist (someone else drops) you will be able to enroll in the lecture (CMPE-012)

- There will be attrition in the class (especially due to lab requirements)
- If you are on the waitlist, you are in CANVAS and on PIAZZA already
- Do the work, keep up with the class.
- You will come off the waitlist in order as others drop out
The Team(11,17),(994,984)

• Instructors
  – Gabriel Hugh Elkaim (elkaim+cmpe012@soe.ucsc.edu)

• Teaching Assistants
  – Ehsan Hemmati
  – Corey Ibanez
  – Maxwell Lichtenstein
  – Michael Powell
  – Carlos Ramirez
Who to Contact for Questions:

• Post your questions onto Piazza:

• If you need to keep the information private, use the “private post” option and only the instructional staff will see it.

• For emergency communication to the instructor, email to: [elkaim+cmpe012@soe.ucsc.edu](mailto:elkaim+cmpe012@soe.ucsc.edu)
Communications Etiquette

• We are always available to help if needed.
• TA’s and Instructor monitor Piazza constantly
• Make your subject descriptive, something generic doesn’t help very much and does not allow the staff to prioritize well
• “Help” is not generally descriptive
• Be polite and professional

Gabriel Hugh Elkaim

CMPE-012/L
UCSC MSI Tutoring Service

• Free tutoring program hosted by UCSC to give students more help in their courses.

• CE12 MSI tutor and small section:
  – Gavin Chen (ghchen@ucsc.edu) - Learning Assistant
  – Tangni Wang (twang63@ucsc.edu) – LSS Tutor
Our online presence

Our online presence...

- Canvas [https://canvas.ucsc.edu/](https://canvas.ucsc.edu/)
  - Assignments - Getting and submitting

- Online forum (Piazza)
  - Main forum for communication
Textbooks


What we will cover in this class

- History and Introduction
- Numbering Systems
- Binary Numbers
- Data Representation
- Digital Logic
- Digital Logic Gates in Transistors
- Digital Logic Structures
- ALU Computations
- Computing Overview
- MIPS ISA
- Intro MIPS Programming
- Arrays and Stacks
- Function Calls / Macros
- MIPS Instruction Decoding
- MIPS Architecture
- IO and Exceptions / Traps / Interrupts
- Number systems revisited
- Fractional Binary
- Floating Point Representation
- Floating Point Arithmetic
- Sequential Logic / Boolean Algebra
What we will NOT cover in this class

- Hardware design (CMPE 100/110)
- Extensive C coding (CMPE 13)
- Software engineering (many)
- Algorithms (CS 101)

This class is intended to be a bottom to top overview of computer systems. Other classes will cover material in greater details.
Course work

• Class (CMPE12) Requirements
  – Attending lectures is highly recommended
  – Midterm and Final

• Lab (CMPE12L) Requirements
  – Going to your lab section meetings for help
  – Weekly/Semi-weekly lab assignments
    • Posted online, submitted online via commit ID
    • Lab README.txt as text file
Lab work

• Part 0: Basic Lab Submittal and Programming
• Part 1: Logic design with Logic Simulator
• Part 2: Programming assignments in MIPS

• Each and every lab requires a MINIMUM SUBMISSION REQUIREMENTS in order to pass the class
Lab Rules

• Each lab assignment consists of two things
  – Lab work (code, design file, etc.)
  – Lab write-up
• Lab assignment score = code + write-up
• Assignments are turned in through Git
• Must submit both the lab code and the write-up
• You have 72 hours of grace period to use as you see fit
Late Policy

- Each day an assignment is late the max possible percentage is dropped. This is as follows:

<table>
<thead>
<tr>
<th>Days Late</th>
<th>Max Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>1</td>
<td>95%</td>
</tr>
<tr>
<td>2</td>
<td>85%</td>
</tr>
<tr>
<td>3</td>
<td>70%</td>
</tr>
<tr>
<td>4</td>
<td>50%</td>
</tr>
<tr>
<td>5</td>
<td>Not accepted any longer</td>
</tr>
</tbody>
</table>

No penalty when using grace period after
Attendance

- Highly recommended for the class
- Lab Attendance
  - Easier to get through the labs with help
  - TA/Tutors will be available then
  - Labs are not designed to be done within your lab section
    - Expect to spend much more time working outside of section than in it
Grades for CMPE12/L

- 50% Labs
- 25% Midterm
- 25% Final

- MINIMUM SUBMISSION REQUIREMENTS to pass the class (each and every lab)
- Cannot make up missed exams
- Same Grade for the Lab as the class
- You must score at least 50% on Labs (averaged), Midterm and Final (averaged) to pass the class.
- This is a necessary but not sufficient condition for passing.
DRC Requests

- Students with special needs should refer to the Disability Resource Center (DRC)
- Notify us within first week of class
- We will accommodate your needs
- Confirm with us and SOE office at least 1 week before each exam so we may try to accommodate you needs
Academic Dishonesty (Cheating)

- Cheating is presenting someone else’s work as your own
- Anyone caught cheating will immediately fail the class and the lab, and be reported to their college
- Copying each other’s code is never acceptable.

Don’t do it—not worth it.
Lab 0

- Lab 0 – Submitting your work using GIT
- This is a very simple lab. You should be able to complete it within a few hours.
- If you cannot complete this, you are most likely not ready to take this class.
- Lab 0 is due FRIDAY at midnight (11:59PM)
Why Take CMPE-012/L

• Understanding the “machine model” of how a computer computes makes you a better coder.
  – You understand the limitations of the computational model
  – Some things you can do to take advantage of the hardware
  – Having knowledge of the low-level allows you to better code at a high level
  – You waste less time

• “Time is our most valuable nonrenewable resource, and if we want to treat it with respect, we need to set priorities.” — Albert-László Barabási
History of Computers
The History of Computers

The history of computers is interesting (or should be if you are in this class) and relevant to our professional lives.
The First Computer Hardware

Charles Babbage, born 1791
- Father of the computer
  - 1830 Difference engine – used mechanical power
    - Calculated mathematical tables
    - Smallest imperfections caused errors
    - Funded by the British government
- Funding was pulled, even his colleagues thought it wouldn't work
  - Conceived of analytical engine to perform many types of calculations
  - His son built a model of the machine
  - Working version finally built 1991
The First Programmer

Ada, the countess of Lovelace

- Mother of computer programming – the first programmer!
- A gifted mathematician
  - She helped develop instructions for computations on the analytical engine.
  - Saw Babbage's theoretical approach as workable
History of Computers

The First Electrical computer

1890 Herman Hollerith
• Able to count the census in 6 weeks rather than 7 years
  – Used Jacquard’s punch cards
  • Sorted into bins
    – Country

• Developed in 1804 by a French silk weaver Joseph Marie Jacquard

Electrical power
Tabulating machine company merged into IBM in 1924
History of Computers

Aiken, Zuse, Atanasoff, Berry

- 1936 – Harvard graduate student Howard Aiken began thinking of modern equivalent of analytical engine...
- 1939 Germany – Konrad Zuse completed first programmable, general-purpose calculating device to solve mathematical problems
- 1939 – Iowa State Professor John Atanasoff developed the first electronic digital computer, the Atanasoff-Berry Computer (ABC)

Paper was in short supply during war, used film tape
1944 Harvard professor Howard Aiken completed the Mark I
- Assistant Grace Hopper
  - Developed compiler for the computer
- 8 feet high, 55 feet long steel and glass
  - Used noisy electromechanical relays
  - 5–6 times faster than a person
  - Not very efficient
First Computer “Bug”

Found on the 9th of September, 1945, by Grace Murray Hopper while she was working on the Harvard University Mark II Aiken Relay Calculator (a primitive computer).

Coined term “debug”.

Photo # NH 96566-KN  First Computer “Bug”, 1945
ENIAC, UNIVAC by John Machly & J Presper

WWII – **ENIAC** Electronic Numerical Integrator and Computer
  - Based on the Atanasoff-Berry Computer
  - Used to calculate trajectory tables for artillery
  - First general-purpose computer

- June 14, 1951 – **UNIVAC 1** – Universal Automatic Computer
  - First general-purpose commercial computer
Von Neumann Architecture
First Draft of a Report on the EDVAC, June 30, 1945

- CPU
- ALU
- Control
- Memory
- I/O
- We still use this model today!

John’s Los Alamos badge
Four generations of computers

1. 1951–1958 Vacuum Tube
   - About the size of light bulbs
   - Thousands of them
   - Is the bug a problem with tube or program?
   - Machine code and punch cards

2. 1959–1964 Transistor
   - Transfers electronic signals across resistor
   - Assembly language
   - 1954 – FORTRAN
   - 1959 – COBOL

Transistor - mighty mite of electronics
History of Computers

Four generations of computers

   - Complete electronic circuit on a small chip of silicon
   - Silicon is a semiconductor - will transmit electrical signal when specific chemical impurities are introduced to lattice structure.
   - IBM 360 series of IBM
   - First time small and medium businesses could afford a computer.
   - Unbundle software – sell software separately
   - Birth of software industry

4. 1971–present Microprocessor (VLSI)
   - Extension of third generation
   - Get specialized chips for memory and logic
The Next Generation?

- The next generation is upon us and you are seeing it in your daily lives. I call it the **SOC – System-on-a-Chip** generation.
  - Put everything on a single chip
    - CPU
    - GPU
    - Memory (or at least part of it)
    - IO
  - Enables very low power with high performance
    - Smart phones, tablets, etc.
  - Also hear it called **Ubiquitous Computing**
    - Computing everywhere – Internet of Things (IoT)
History Summary

- Knowing something about the evolution of computers is helpful to understanding why things are the way they are now
- Computing devices have been around a long time
- Digital computers are fairly new
- Rate of improvement and growth is amazing, Moore’s Law