CMSC131

Welcome to the Class, the Department, the University

Some terminology to consider…

Computer Science: a field that contains both theoretical and practical elements, with heavy overlap between them as the theoretical work influences practical applications

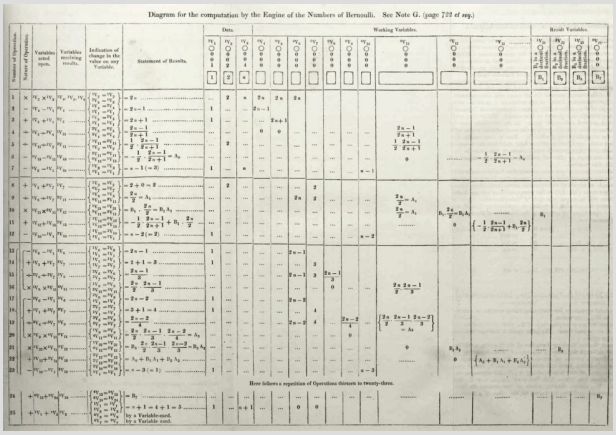
Software Engineering: part of the world of computer science focused on the design, implementation, deployment, and maintenance of software

Computer Programming: this can be seen as the part of software engineering where you go from the description of a part of a computing problem to the code that accomplishes it

Computing: a term that has many meanings and contexts, ranging from designing and building software to using it to accomplish a goal

The first computer program…

The first computer program (written by Ada Lovelace in 1842 to calculate a sequence of Bernoulli numbers) ***pre-dates*** the first device that would have been able to execute it!



What do computer scientists do?

The field of computer science is a mixture of both scientific exploration and practical applications.

Examples include:

- Computational Thinking and Problem Solving
- Tool Building
- Technique Design
- Prototyping and Implementation
- Software Testing

A computer scientist can be part of a broader team working to "solve a problem" of some sort.

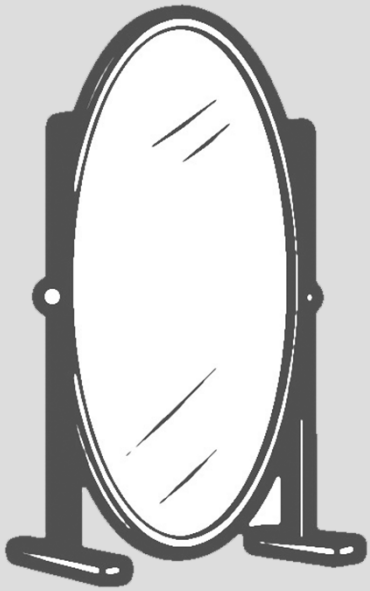
Why quotes?

On the previous slide I have “solve a problem” in quotes. Why? Some well-known examples in computing clearly solved an existing problem:

- Word processors solved the problems with correcting typos, updating existing documents, and mixing text and images.
- Search engines solved the problem of there not being an organizational framework for the web.

However, before they existed, would people say there was a clear “problem” to be solved that lead to Facebook, Twitter, Pokemon GO, etc.?

Who can be a CS/CE major/minor?



You!

- Realize that different students will need to invest different amounts of time in their courses to succeed.
- Realize that in the end not everyone wants to be a CS/CE student, but if you are here you should keep an open mind to it and do your best.
- If you decide it’s not worth the time investment or isn’t where you want to spend your time, don’t feel bad but make it a decision you make.

Let's work through a problem…

You are given an integer. Similar to in algebra class, let's refer to it by a variable name like ***n***.

We want to create a routine that will return a value of **true** if that ***n*** value is within 10 of either 100 or 200.

As a human computer, what would your algorithm be?

Once we have talked it through, we can explore this in Java at codingbat.com/prob/p184004

One possible solution

```
public boolean nearHundred(int n) {
int differenceFrom100 = 100 - n;
int differenceFrom200 = 200 - n;
   return Math.abs(differenceFrom100) <= 10
                  ||
          Math.abs(differenceFrom200) <= 10;
}
```

What is CMSC131 about?

You will be practicing computational thinking and applying it to solving problems while learning about tools that programming languages provide to solve problems / accomplish tasks.

Implementing programs of different sizes and complexity using Java, which is often a cyclical process that includes:

- understanding project specifications
- outlining the approach to solving a problem or accomplishing a task
- implementing program modules based on the above
  - **testing and fixing these modules as you build them is an important part of implementation**

"Outsiders" don't get it sometimes…

In the early 2000s, a speaker from the business management side of things told a group at a workshop that one set of people wrote code and a different set of people fix the bugs in it.

- That's not quite how it works…

Precision Matters!

When you write a paper, if you have a single spelling error in each paragraph the paper can probably still be understood.

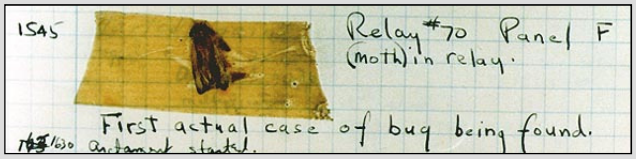
If a report for a class covers several related topics, and you have a slight misunderstanding expresses in each you'll likely still pass.

In a computer program, if each module has a single logic error or typo, it is actually quite possible that it accomplishes none of its goals correctly!

- **Unlike papers and reports, we provide a way to allow you to pre-check much of your work as you go!**

Bugs

The notion of a "bug" dates back at least as far as Thomas Edison, but the most well-known bug might be this one, found by Grace Hopper:



If you write code that doesn't work in certain cases, you will likely be the one fixing them.

- However, on a large enough and/or long enough project, you might "inherit" the code of another person, and if bugs are found at that point you will need to fix or deal with them…

Syllabus / Logistics

Let's talk about…

- Slides: read them, PDF might have additional slides
- Exams: prepare for them via projects, labs, etc.
- Projects: start as assigned, work incrementally
- Lab quizzes and coding exercises: weekly
- Clickers: consider your answers
- Office hours: posted on class site
- When you **can** and when you **cannot** work with others http://cs.umd.edu/class/fall2019/cmsc131-010X/OpenClosedPolicy.shtml

"College is not High School"

While coursework (projects, labs, etc.) will have deadlines and checkpoints, you are likely going to find yourself more responsible for managing your time on a day-to-day basis than in a high school environment.

If you have a question about the material, ask us. It might be difficult to in class due to the number of students, but we have many office hours.

You are surrounded by students who were able to get admitted to UMD! This is a new crowd and potentially a different level of peers than you are used to.

"Computer Science isn't like ___"

Different majors have different types of work and different levels of requirements.

While a grade of C- or better is required to advance to the next course(s) in the major, aiming for an A or B is a wiser goal since our sequence of courses build on one another and assume a high degree of understanding of material from the previous course.

One of the goals of the earlier classes is to develop good habits that will make the later courses more manageable.

CMSC 131 is just the beginning…

The foundational programming sequence is CMSC 131, 132, 216, 330.

- Programming languages, tools, tricks, etc. in Java and C as well as learning about how to learn / pick new programming languages.

There is also a more mathematically oriented sequence of MATH 140, 141, and CMSC 250, 351.

- Math skills and domains related to computer science, how to formally prove things, and applications of those to topics in computer science.

The upper-level courses where you get breadth and depth in different parts of the field.

Upper Level Courses

There are a wide variety of courses across many of the topics within Computer Science.

For example…

Systems Architecture, Operating Systems, Networks, Network Security, Data Structures, Artificial Intelligence, Machine Learning, Bioinformatics, Databases, Computer Vision, Graphics, Compilers, Concurrency, Software Engineering, Human-Computer Interaction, Mobile Programming, Algorithm Design and Analysis, Cryptology, Numerical Analysis, …

Can you guess your own hat color?

Imagine that you and your neighbor here in class were wearing hats that someone else put on your head and that you could not see.

Assume there are only two hat colors possible, red and blue and that you would each need to make a guess at your own hat's color at the same instant as your neighbor made their guess.

Could you come up with a plan where at least one of you guess right?

Project Management

Start working on the projects when they are assigned.

- Read the entire description to see where you will be going. Work on it incrementally.
- Work to complete the project ***and*** to understand the concepts they utilize.
- Keep time in your schedule to come to office hours if needed and includes "wiggle room" for unexpected problems in or out of this class.

Try to avoid overloading yourself with obligations; whether academic, employment, or social.

Practice, Practice, Practice

If you get help solving a coding problem in office hours, it is often a good idea to make a "fresh" attempt at that section of code again a few days later to review the material.

Think of studying for the exams and preparing for projects as a continuous process. After a new idea is presented in class:

- review it after class, perhaps with a friend
- experiment and explore it with one of the posted code samples

"Show Up" physically and mentally

Plan to invest in this course in your time, energy, and effort and to attend all of your lecture and discussion sections.

- Schedule your life so you are "ready" for class or lab (not half-asleep, hung-over, distracted, etc.)
- Check the class website frequently, perhaps daily, for reminders or new code examples or slides.
- Avoid multi-tasking if one of those tasks is related to learning or practicing material for this class.
- Understand that with your projects, "Due Wednesday" ***does not mean*** "Do Wednesday!"

Learning and Assistance

Remember…

- The goal of most activities related to this class is learning about computational thinking as well as implementation. We learn by doing. Making and working through mistakes are a natural part of the learning processing.
- Before going to office hours for help, you should have really thought about the problem and made honest attempts to work through it by reviewing it as well as examples from class.
- You cannot talk about projects with fellow students or try to find code others have written, but you can talk about them with ***us*** in office hours, or with a rubber duck.