## Conclusion

**Please take the time to complete the online course evaluation for all your Engineering Courses. For this course, completion of the evaluation is required and carries a weight of 1.**

**FINAL**

**Monday, 14 Dec, 7:00-9:30 PM, Baker Lab 200**

**8 review sessions next week.**
**See handout about the final for details.**

**You should have emailed Maria Witlox if you have a conflict!**

0

---

## Punch cards

**Jacquard loom**

Mechanical loom invented by Joseph Marie Jacquard in 1801.

Used the holes punched in pasteboard punch cards to control the weavinng of patterns in fabric.

Punch card corresponds to one row of the design.

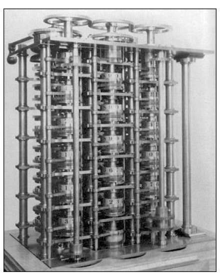
Based on earlier invention by French mechanic Falcon in 1728.

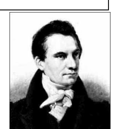
1

---

Charles Babbage designed a “difference engine” in 1822

Compute mathematical tables for log, sin, cos, other trigonometric functions.





The mathematicians doing the calculations were called computers

2

---

Oxford English Dictionary, 1971

Computer: one who computes; a calculator, rekoner. spec. a person employed to make calculations in an observatory, in surveying. etc.

1664: Sir T. Browne. The calendars of these computers.

1704. T. Swift. A very skillful computer.

1744. Walpole. Told by some nice computers of national glory.

1855. Brewster Newton. To pay the expenses of a computer for reducing his observations.

The mathematicians doing the calculations were called computers

3

---

Charles Babbage planned to use cards to store programs in his Analytical engine. (First designs of real computers, middle 1800s until his death in 1871.)

First programmer was Ada Lovelace, daughter of poet Lord Byron.

Privately schooled in math. One tutor was Augustus De Morgan.

**The Right Honourable Augusta Ada, Countess of Lovelace.**

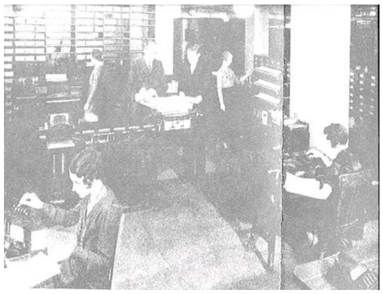
---

**Herman Hollerith.**
His tabulating machines used in compiling the 1890 Census.
Hollerith's patents were acquired by the Computing-Tabulating-Recording Co.
Later became IBM.

The operator places each card in the reader, pulls down a lever, and removes the card after each punched hole is counted.

Hollerith 1890 Census Tabulator

Computers (mainly women), calculating the US census



6

**History of computers**

**1935-38.** Konrad Zuse - **Z1 Computer**

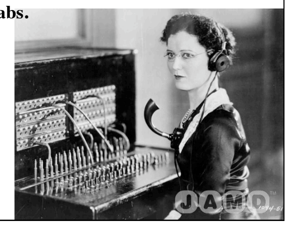
**1935-39.** John Atanasoff and Berry (grad student). Iowa State

**1944.** Howard Aiken & Grace Hopper **Harvard Mark I Computer**

**1946.** John Presper Eckert & John W. Mauchly **ENIAC 1 Computer** 20,000 vacuum tubes later ...

**1947-48 The Transistor, at Bell-labs.**

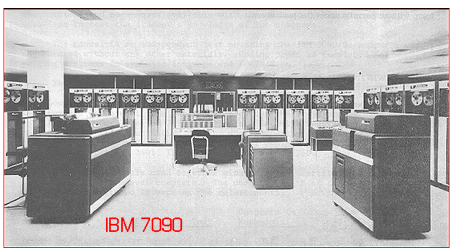
**1953. IBM, the IBM 701.**

How did Gries get into Computer Science?

1959. Took his only computer course. Senior, Queens College.

1960. Mathematician-programmer at the US Naval Weapons Lab in Dahlgren, Virginia.




8

How did Gries get into Computer Science?

1959. Took his only computer course. Senior, Queens College.

1960. Mathematician-programmer at the US Naval Weapons Lab in Dahlgren, Virginia.

Programmed in Fortran and IBM 7090 assembly language

```
         CLI   SEX,'M'       Male?
         BNO   IS_FEM        If not, branch around
         L     7,MALES       Load MALES into register 7;
         LA    7,1(,7)       add 1;
         ST    7,MALES       and store the result
         B     GO_ON         Finished with this portion
IS_FEM   L     7,FEMALES     If not male, load FEMALES into register 7;
         LA    7,1(,7)       add 1;
         ST    7,FEMALES     and store
GO_ON    EQU   *
```

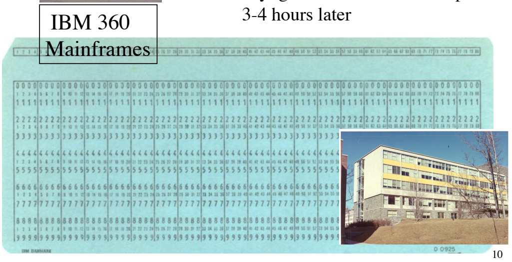
**if** (SEX == 'M') MALES= MALES + 1;
**else** FEMALES= FEMALES + 1;

9

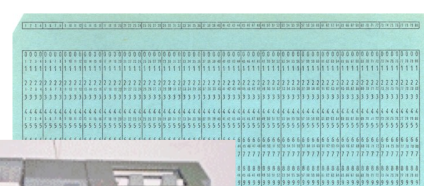
**Late 1960s**

IBM 360 Mainframes

Write programs on IBM "punch cards. Deck of cards making up a program trucked to Langmuir labs by the airport 2-3 times a day; get them back, with output, 3-4 hours later



10

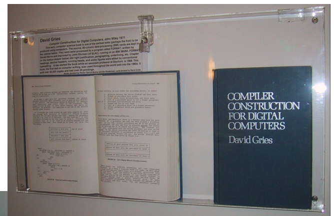


Gries's 1971 compiler construction book was punched on IBM cards. They are now on display at Stanford

http://infolab.stanford.edu/pub/voy/museum/pictures/display/floor5.htm

11

The text for the book was punched onto ~12,000 computer data-processing (IBM) cards. Many figures also created using characters.




Each card: 12 rows of 80 cols. 72 cols for characters of text, 8 cols for identification.

12

**About 1973. BIG STEP FORWARD**

1. Write program on punch cards.
2. Wait in line (20 min) to put cards in card reader in Upson basement
3. Output comes back in 5 minutes

**About 1973. BIG STEP FORWARD**

Switched to using the programming language Pascal, developed by Niklaus Wirth at Stanford.

**About 1979. Teraks**

Prof. Tim Teitelbaum sees opportunity. He and grad student To.. Reps develop "Cornell Program Synthesizer". Year later, Cornell uses Teraks in its prog course.

40 lbs

November 1981, Terak with 56K RAM, one floppy drive: $8,935.

Want 10MB hard drive? $8,000 more

13

**1983-84**

Switched to Macintosh in labs

**1980s**

CS began getting computers on their desks.

**Late 1980s**

Put fifth floor addition on Upson. We made the case that our labs were in our office and therefore we need bigger offices.

**Nowadays**

Everybody has a computer in their office.

14

**Programming languages. Dates approximate**

| Year | Major languages | Teach at Cornell |
|---|---|---|
| 1956's | Fortran | |
| 1960 | Algol, LISP, COBOL | |
| 1965 | PL/I | PL/C (1969) |
| 1970 | C | |
| 1972 | Pascal | |
| 1980's | Smalltalk (object-oriented) | Pascal (1980's) |
| 1980's | (late) C++ | |
| 1996 | Java | C and C++ |
| 1998 | | Java / Matlab |

15

**During 1970s, 1980s, intense research on**
How to prove programs correct,
How to make it all practical,
Methodology for developing algorithms

The way we understand recursive methods is based on that methodology.
Our understanding of and development of loops is based on that methodology.

Throughout, we try to give you thought habits to help you solve programming problems for effectively

Mark Twain: Nothing needs changing so much as the habits of **others**.

16

Throughout, we try to give you thought habits to help you solve programming problems for effectively

Simplicity is key:
Learn not only to simplify, learn not to complify

Don't solve a problem until you know what the problem is

Separate concerns, and focus on one at a time.

Specify methods before writing them

Develop and test incrementally

Read a program at different levels of abstraction

Define variables before using them (e.g. class invariant, loop invariant)

Use methods to avoid duplication, keep program simple

17

### Simplicity and beauty: keys to success

CS has its field of computational complexity. Mine is computational simplicity,

**David Gries**

**CS professor's non-dilemma**

I do so want students to see
beauty and simplicity.
A language used just has to be
one only with that property.
Therefore, and most reasonably,
I will not and do not teach C.

**David Gries**

Inside every large program is a little program just trying to come out. **Tony Hoare**

**Bugs**

Your
testing
shows presence
but never absence

**Admonition**
**a little Grook**

In correctness concerns
one must be immersed.
To use only testing
is simply accursed.

18

### On Science and Engineering

Science explains why things work in full generality by means of calculation and experiment.
Engineering exploits scientific principles to the study of the specification, design, construction, and production of working artifacts, and improvements to both process and design.

Science asks: WHY? Engineering asks: WHY NOT?

19