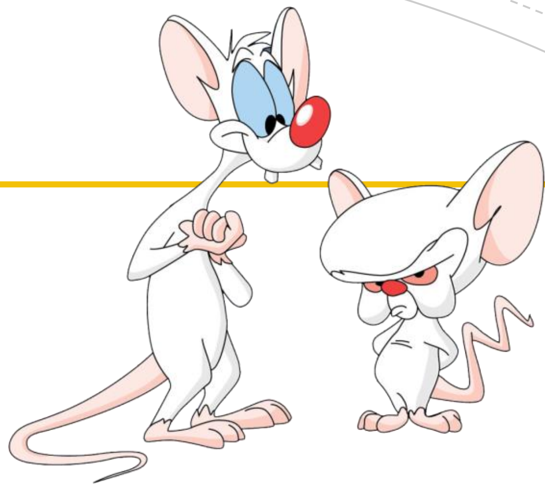
# COMP 202
# Foundations of Programming

Lecture 1: Intro & Binary numbers

Giulia Alberini, Fall 2018

## WHAT ARE WE GOING TO DO TODAY?



- A little about the course
- Start of course material
  - Algorithms
  - Binary numbers

## ABOUT ME

Giulia Alberini

- Faculty Lecturer in SOCS
- PhD in Computer Science from McGill (Cryptography)
- MSc and BSc in Mathematics from University of Padova
- Love to play Ultimate

Contact Info

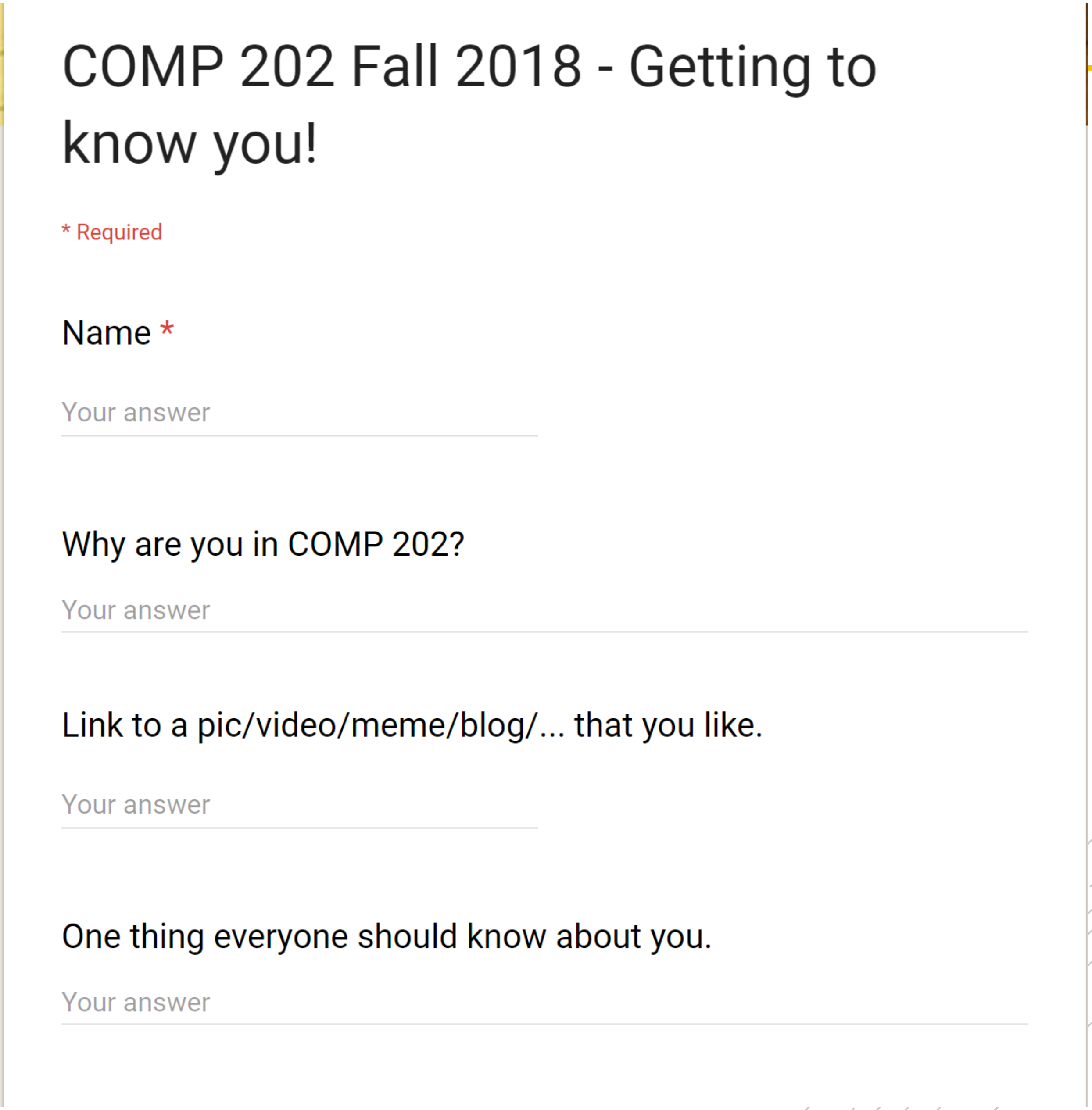
| | |
|---|---|
| Email: | giulia.alberini@mcgill.ca |
| Office: | TBA |
| Office hours: | Thursdays 10am-12:30pm (or by appointment) |

## HOW ABOUT YOU? WHY ARE YOU HERE?

1. This is a required course in my program
2. Is Computer Science for me?
3. Might be useful to learn how to program.
4. I want to write the new app of the year!
5. Couldn’t find my way around

## GETTING TO KNOW YOU!

Please take a couple of minutes to fill out this simple form: https://goo.gl/forms/w03zqn00n0AsDFG72



COMP 202 Fall 2018 - Getting to know you!

* Required

Name *

Your answer

Why are you in COMP 202?

Your answer

Link to a pic/video/meme/blog/... that you like.

Your answer

One thing everyone should know about you.

Your answer

## WHAT IS THIS COURSE ABOUT

- Introduction to computer programming.
- Programming language: Java
- Little or no background needed.
  The ability to **think logically and in an abstract manner** is more important than any college-level mathematical background.

Note:

- This course is NOT about how to use a computer.
- COMP 202 and COMP 208 cannot both be taken for credit
- COMP 202 cannot be taken for credit with or after COMP 250.

## BY THE END YOU'LL BE ABLE TO

- Understand how a computer "thinks"
- Design and describe instructions to perform a task in a way that a computer can understand
- Structure a larger problem into smaller and simpler problems.
- Translate these problems into the right set of instructions in Java
- Build non-trivial programs.
- Learn independently about new programming-language features.
- Pass the course and have fun!

## IS THIS COURSE HARD?

Learning how to program can be very time consuming

- The programs we write need to be “perfect” – computers cannot “fill in the blanks”.
- On a good note, we can keep improving our program until it works!
- The hard part is not writing a program, it’s fixing it.

You don’t have to be a genius, you need to be determined! ;)

## HINTS FOR TAKING THIS COURSE

- Practice!

  Whenever you see examples in class or in the textbook try them on your laptop!

- Do the homework.

  Finding a solution on the web is not a substitute for thinking for yourself. You need to engage with the material in order to fully understand the topics and be ready for the tests.

- Ask for help.

  Make sure you don't fall behind. Ask questions in class and go to office hours.

- Don't give up!

  If you cannot follow everything that is going on in the classroom, don't feel discouraged. Some topics need time to sink in.

## STRUCTURE OF THE COURSE

- Week 1 – 7: Fundamentals of programming.
- Week 8 – 11: Intro to Object Oriented Programming
- Week 12 - 13: Some more in-depth/advanced topics.

For more details, see the course Syllabus.

## COURSE COMMUNICATION

- myCourses: http://www.mcgill.ca/lms
- You can find there all information regarding the course: schedule, instructors' and TAs' office hours, announcements, grades, assignments, and so on.
- Please, post your questions about assignments on the myCourses message board so that everyone can see both the questions and the answers.
- You may also answer other students' questions.
  - OK to provide one or two lines of code to illustrate a point
  - NOT ok to provide solution code.

## NOTE ABOUT EMAILS

- If you email me for <u>an urgent</u> matter write URGENT in the subject line.
- If you have problems installing the software needed for the course please go see one of the TAs.
- If you have involved questions please come to see me or the TAs during our office hours.
- Questions about assignments should be posted on myCourses.
- For questions regarding your assignment's grade: first email the TA who graded your assignment.
- Please avoid sending screenshots of your programs and ask us to find the error.

## LECTURES

- 3 hours per week:
  Monday, Wednesday, and Friday 12:35-13:25

- Try to do the assigned readings before the class (you can find all the info on the course syllabus).

- Make sure to ask questions if you don't understand.

- Laptops can be useful in order to try out the examples we see in class, but please keep focused on the lecture.

## TEXTBOOK

- Recommended Textbook:
  "Think Java. How to Think Like a Computer Scientist", 6th Edition.
  Allen B. Downey and Chris Mayeld
  Available free at:
  http://greenteapress.com/thinkjava6/thinkjava.pdf

For other references see the course syllabus.

## ASSIGNMENTS (35%)

- 5 assignments
- Very important to complete ALL the assignments.
- The assignments must represent your personal effort!
- **Start them early!**
- Late policy:
  - 10% will be deducted for each day for which they are late (including weekend days and holidays).
  - Assignments submitted more than 2 days after the deadline will not be accepted.

## MIDTERM (20%)

When: October 16th , 6pm-9pm

Room assignments will be announced in class and posted on the course webpage.

- Highly recommended you attend.
- If not, your midterm grade will be allocated to the final
- **No make up exam**.

## GRADING SCHEME

Your final grade in the course is calculated by taking (automatically) the maximum of the following two options:

Assignments: 35%
Midterm: 20%
Final: 45%

OR

Assignments: 35%
Final: 65%

## PLAGIARISM

- Don’t do it!

- There are severe consequences that are not worth the risk.
  You can read all about it here: www.mcgill.ca/integrity/

- It is easy to detect similarities in your assignments. We will be using software detection tools. It would take you more effort to try to conceal the fact that an assignment was copied than actually do it yourself.

## SOCS COMPUTER LABS

- Any student registered to this course may use these facilities to do their work.
- Location: 3rd floor of the Trottier Building.
- For more information refer to: https://www.cs.mcgill.ca/about/facilities/

## REQUIRED SOFTWARE

You are encouraged to use your personal computers or laptop to complete the course work. You will need the following software for this course:

- **Required**: Java Development Kit (JDK).
  The JDK is installed on the computers in the SOCS labs. Follow instructions on the syllabus to download it and install it on your computer.
- **Highly Recommended**: Dr. Java
  You can use any plain-text editors to write your programs, but I recommend a simple IDE (integrated development environment) called Dr. Java. It provides an editor that allows you to write your program, as well as commands to compile and run it.
- **Optional**: Other IDEs such as Eclipse.

## ASSISTANCE

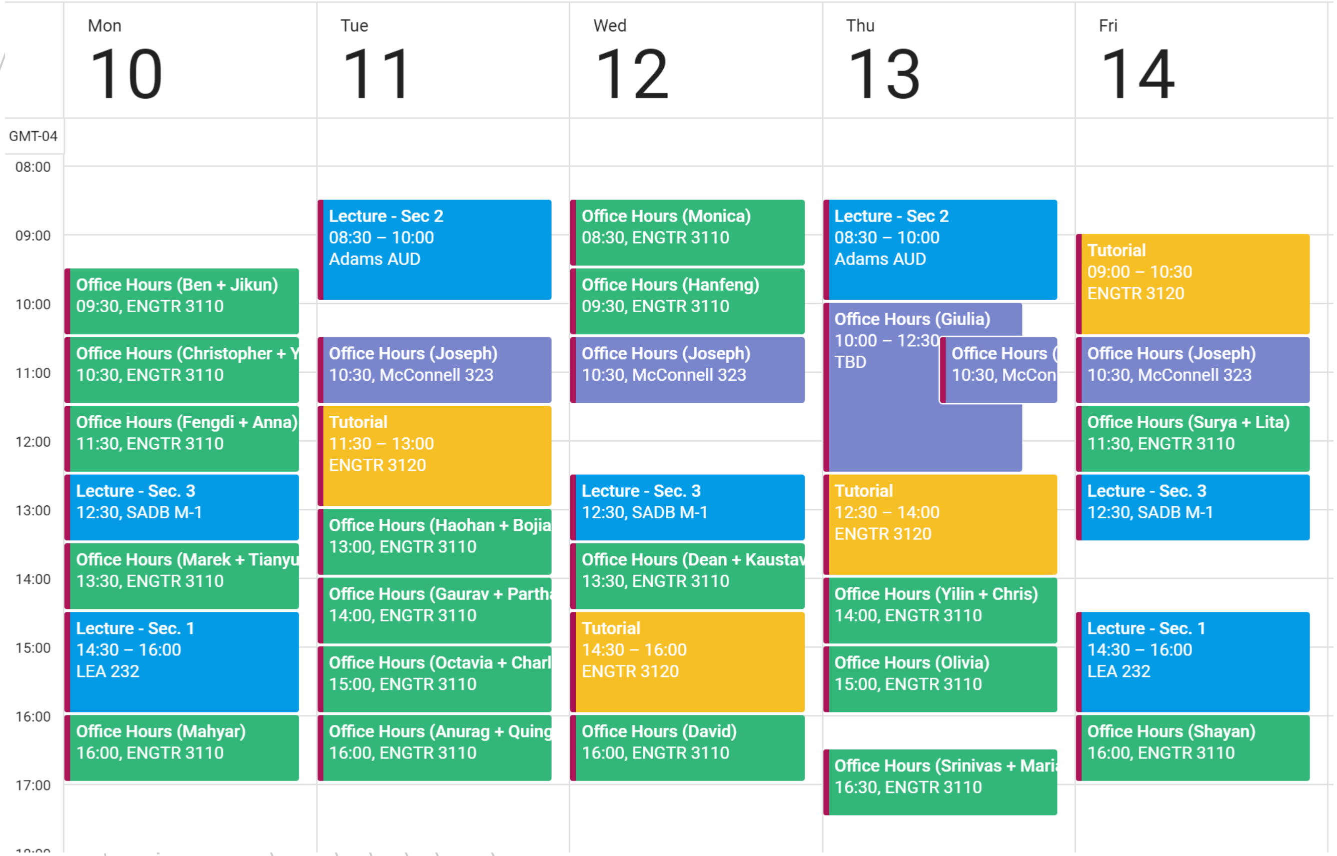
TAs

- Feng
- Bojia
- Haohan
- Jikun
- Anurag
- Octavia
- David
- Mahyar
- Hanfeng
- Tianyu
- Surya
- Srinivas
- Qinglong
- Monica
- Marek
- Shayan
- Yuzhou
- Lita
- Ben
- Gaurav

TEAM Mentors

- Christopher
- Anna
- Partha
- Chris
- Maria
- Dean
- Yilin
- Kaustav
- Charlotte
- Olivia

# ASSISTANCE – GOOGLE CALENDAR



GMT-04

Mon 10

- Office Hours (Ben + Jikun) 09:30, ENGTR 3110
- Office Hours (Christopher + Y 10:30, ENGTR 3110
- Office Hours (Fengdi + Anna) 11:30, ENGTR 3110
- Lecture - Sec. 3 12:30, SADB M-1
- Office Hours (Marek + Tianyu 13:30, ENGTR 3110
- Lecture - Sec. 1 14:30 – 16:00 LEA 232
- Office Hours (Mahyar) 16:00, ENGTR 3110

Tue 11

- Lecture - Sec 2 08:30 – 10:00 Adams AUD
- Office Hours (Joseph) 10:30, McConnell 323
- Tutorial 11:30 – 13:00 ENGTR 3120
- Office Hours (Haohan + Bojia 13:00, ENGTR 3110
- Office Hours (Gaurav + Parth 14:00, ENGTR 3110
- Office Hours (Octavia + Charl 15:00, ENGTR 3110
- Office Hours (Anurag + Quing 16:00, ENGTR 3110

Wed 12

- Office Hours (Monica) 08:30, ENGTR 3110
- Office Hours (Hanfeng) 09:30, ENGTR 3110
- Office Hours (Joseph) 10:30, McConnell 323
- Lecture - Sec. 3 12:30, SADB M-1
- Office Hours (Dean + Kaustav 13:30, ENGTR 3110
- Tutorial 14:30 – 16:00 ENGTR 3120
- Office Hours (David) 16:00, ENGTR 3110

Thu 13

- Lecture - Sec 2 08:30 – 10:00 Adams AUD
- Office Hours (Giulia) 10:00 – 12:30 TBD
- Office Hours ( 10:30, McCon
- Tutorial 12:30 – 14:00 ENGTR 3120
- Office Hours (Yilin + Chris) 14:00, ENGTR 3110
- Office Hours (Olivia) 15:00, ENGTR 3110
- Office Hours (Srinivas + Mari 16:30, ENGTR 3110

Fri 14

- Tutorial 09:00 – 10:30 ENGTR 3120
- Office Hours (Joseph) 10:30, McConnell 323
- Office Hours (Surya + Lita) 11:30, ENGTR 3110
- Lecture - Sec. 3 12:30, SADB M-1
- Lecture - Sec. 1 14:30 – 16:00 LEA 232
- Office Hours (Shayan) 16:00, ENGTR 3110

- Link to the Google Calendar [https://calendar.google.com/calendar?cid=b3U2bmwwaGM1dmxqaGo1dWVubmI2b2pvZmtAZ3JvdXAuY2FsZW5kYXIuZ29vZ2xlLmNvbQ](https://calendar.google.com/calendar?cid=b3U2bmwwaGM1dmxqaGo1dWVubmI2b2pvZmtAZ3JvdXAuY2FsZW5kYXIuZ29vZ2xlLmNvbQ)
- Locations:
  - TAs/TEAM mentors office hours take place in Trottier 3110
  - Tutorials take place in Trottier 3120.
- Every week:
  - 3hrs of Lectures
  - 23.5hrs of Office Hours
  - 6hrs of Tutorials

## TUTORIALS

- Throughout the term, there will be several (optional) tutorials.
- Designed to help you with the material and assignments.
- It's a chance to ask questions in a smaller environment than lectures.
- Information will be posted on myCourses.
- It is not necessary to register for tutorials.

## TUTORIALS – "GET STARTED"

Designed to help you install the required software for the course.

- Session 1:
  - When: Thursday Sept. 6th, 12:30pm - 2:00pm
  - Where: ENGTR 3120
- Session 2:
  - When: Friday Sept. 7th, 9:00am - 10:30am
  - Where: ENGTR 3120

## ANY QUESTIONS?

## COMING UP NEXT

- What is computer science?
- What is programming?
- What are algorithms?
- Binary numbers

## WHAT IS COMPUTER SCIENCE?

- Computer science is a discipline that spans theory and practice.
- It requires thinking both in abstract terms and in concrete terms.
- Computer science can be seen as a science of problem solving. A computer scientist
  - must be able to model and analyze problems
  - design solutions, and
  - verify that they are correct.

## WHAT IS PROGRAMMING?

Programming can be thought as a 3 step process:

1. Think of a solution
2. Translate it into a sequence of instructions that specifies how to perform a computation (i.e., a program). (*coding*)
3. Check to see if it is correct (*testing*) and fix any problems (*debugging*).

When a problem is complex, part of programming is breaking down such problem into smaller and simpler subtasks that are easy to solve.

## WHAT IS COMPUTER PROGRAMMING?

[https://youtu.be/IoPx_rSicrM](https://youtu.be/IoPx_rSicrM)

## KEY POINTS!

- It’s more about the process of breaking down big problems into smaller ones than coming up with complicated algorithms.
- You don’t have to be a genius!
- It’s for everyone!
- Computer programming is all over! There are many different kind of CS → you can apply programming to what interests you!
- You can be a wizard, a super hero, and a rockstar!!!

## WHAT ARE ALGORITHMS?

- An algorithm is a sequence of steps that specifies how to solve a problem.

- Using the terminology we have just learned, we can say that computer science is the science of algorithms.

## EXAMPLES

- A recipe to cook your favourite pie.
- The list of instructions you need to follow to install Fortnite.
- The moves to solve a Rubik’s cube.
- The procedure we follow to convert decimal (base 10) numbers into binary (base 2) numbers.

## NUMBER BASES
## and
## BINARY NUMBERS

## HOW DOES A COMPUTER WORK?

- Computers are made of wires. Current (electricity) can either pass through each wire, or not.
- It is a huge amount of light switches that can be turned on and off.

## ON AND OFF

Suppose I am a (terrible) magician. I leave the room and my assistant can then flip a coin. Beside the audience, the only thing in the room is a light with a switch. How can the assistant communicate the result of the coin toss to me? (Note that we can agree on a code beforehand)

- We can decide in advance that 'on' is code for heads and 'off' is code for tails.

## ON AND OFF

Consider a new trick: there are three cups lined up on a table. When I leave, my assistant puts a ball under one of the cups. How many light switches/what code can we now use to communicate this information?

## SOLUTION 1

Have three switches:

- switch 1 on → the left cup
- switch 2 on → the middle cup
- switch 3 on → the right cup

- Can we do better?

## SOLUTION 2

Have two switches:

- both off → the left cup
- switch 1 on only → the middle cup
- switch 2 on only → the right cup
- both off → nothing

- How many switches would we need for a 6 sided die?

## BINARY INTRO

- We would like to represent numbers (and other data) in our computers
- But we saw that computer are made of wires and switches
- We need to represent numbers using just on and off.
- To do so, we use the ‘base 2’ system – called binary
  - Off = 0
  - On = 1

## BASE 10

When we refer to a decimal (base 10) number, like $5764$, we are referring to the value obtained by carrying out the following addition:

$$5000 + 700 + 60 + 4$$

That is, we add together:

- Ones: 4
- Tens: 6
- Hundreds: 7
- Thousands: 5

## EXPONENTS AND POWERS

- Reminder

  $10^3 = 10 \cdot 10 \cdot 10 =$ 'ten raised to the power of 3'
  $n^0 = 1$ for any number $n$.

- So, using the exponential notation, we can write

$$5764 = 5 \cdot 10^3 + 7 \cdot 10^2 + 6 \cdot 10^1 + 4 \cdot 10^0$$

## BASE 10

$$5764 = 5 \cdot 10^3 + 7 \cdot 10^2 + 6 \cdot 10^1 + 4 \cdot 10^0$$

NOTE:

- The digits of the number correspond to the coefficients of the powers of ten
- The position of the digit in the number determines to which power it is associated.

In a similar way, we can write numbers in other bases (beside 10):

- We use the digits that correspond to the coefficients on the corresponding powers (of the given base)

## NOTATION

We write

$$n_b$$

to denote that the number $n$ is written in base $b$.

## EXAMPLE

- What is the value of the base 5 number $132_5$ ?
- We need to first compute the corresponding powers of $5$:

$$5^0 = 1$$
$$5^1 = 5$$
$$5^2 = 25$$

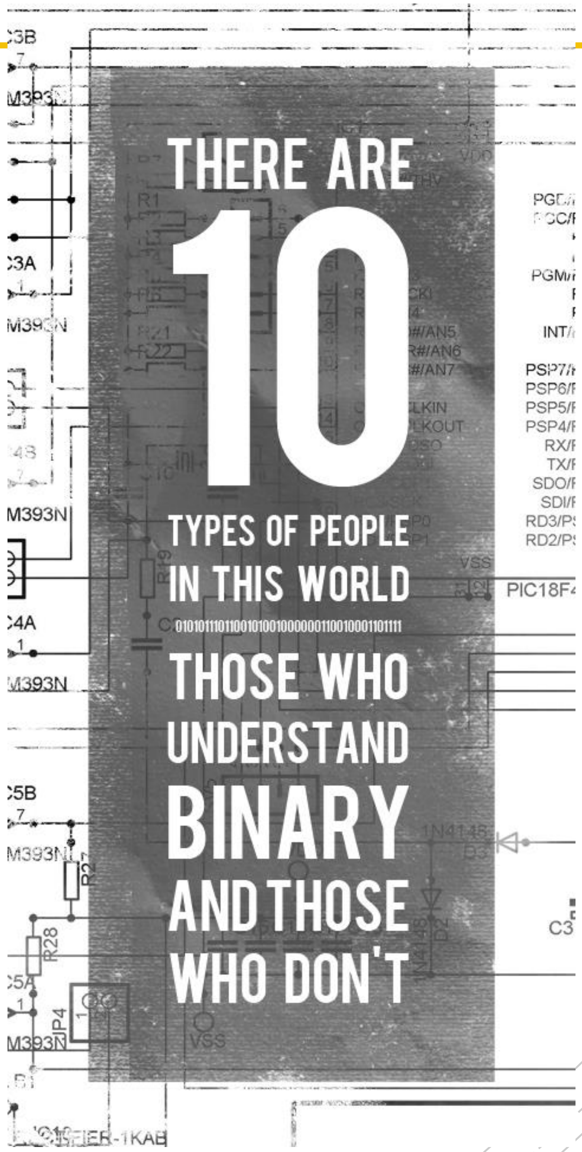
- Now we multiply them by the corresponding digit and sum the results together:

$$1 \cdot 5^2 + 3 \cdot 5^1 + 2 \cdot 5^0 = 25 + 15 + 2 = 42$$

## BASES

Important representations / bases:

- Decimal / base 10
- Binary / base 2
- Octal / base 8
- Hexadecimal / base 16
- Although in everyday life we use decimal notation to express integers, it is often convenient to use bases other than 10. In particular, computers work in binary numbers.



## COUNTING IN BINARY

| | | |
|---|---|---|
| $0_{10}$ | $0_2$ | $0 \cdot 2^0$ |
| $1_{10}$ | $1_2$ | $1 \cdot 2^0$ |
| $2_{10}$ | $10_2$ | $1 \cdot 2^1 + 0 \cdot 2^0$ |
| $3_{10}$ | $11_2$ | $1 \cdot 2^1 + 1 \cdot 2^0$ |
| $4_{10}$ | $100_2$ | $1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$ |
| $5_{10}$ | $101_2$ | $1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$ |
| $6_{10}$ | $110_2$ | $1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$ |
| $7_{10}$ | $111_2$ | $1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$ |

## BINARY TO DECIMAL

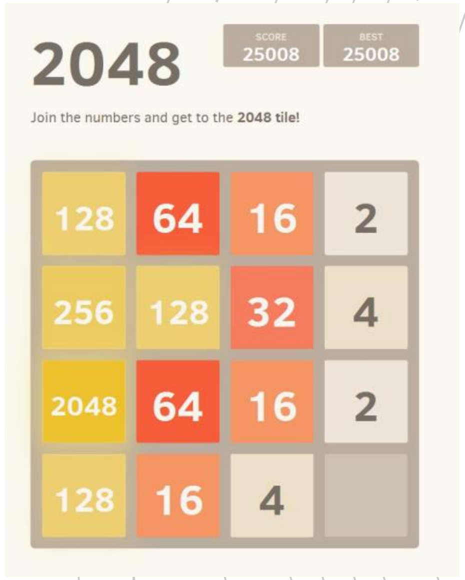
- In general, we know that if we can use the following binary representation $(a_k a_{k-1} \ldots a_1 a_0)_2$ for a number, then its value is

$$n = a_k 2^k + a_{k-1} 2^{k-1} + \cdots + a_1 2^1 + a_0 2^0$$

- NOTE: $a_i = 0$ or $1$.

  Thus only the terms with the $a_i = 1$ will be left to sum!

## BINARY TO DECIMAL - ALGORITHM



Given a binary number, do:

- Compute the powers of 2 needed (as many as the binary digits in the number)
- Identify the powers associated to the digits equal to 1
- Sum them all together.

| $x$ | $2^x$ |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 4 |
| 3 | 8 |
| 4 | 16 |
| 5 | 32 |
| 6 | 64 |
| 7 | 128 |
| 8 | 256 |
| 9 | 512 |
| 10 | 1024 |
| 11 | 2048 |

## BINARY TO DECIMAL – EXAMPLE 1

Convert $\mathbf{11010_2}$ to a decimal number.

- Computer the powers
- Identify the powers associated to the digits equal to 1.
- Sum them together

| | | | | | |
|---|---|---|---|---|---|
| Position/ Exponent | 4 | 3 | 2 | 1 | 0 |
| Powers of 2 | 16 | 8 | 4 | 2 | 1 |
| Digits | **1** | **1** | 0 | **1** | 0 |

| | |
|---|---|
| Output: | $16 + 8 + 2 = \mathbf{26}$ |

## BINARY TO DECIMAL – EXAMPLE 2

Convert $\mathbf{11010101_2}$ to a decimal number.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Position/ Exponent | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Powers of 2 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| Digits | **1** | **1** | 0 | **1** | 0 | **1** | 0 | **1** |
| Output: | $128 + 64 + 16 + 4 + 1 = \mathbf{213}$ | | | | | | | |

## DECIMAL TO DECIMAL

What is $5764_{10}$ in decimal notation (base 10)?

$$\frac{5764}{10} = 576\ R\ \boxed{4}$$

$$\frac{576}{10} = 57\ R\ \boxed{6}$$

$$\frac{57}{10} = 5\ R\ \boxed{7}$$

$$\frac{5}{10} = 0\ R\ \boxed{5}$$

Note that taking the remainders from bottom to top gives us the answer.

## DECIMAL TO BINARY

The easiest way is using parity checks and divisions by $2$. What is $13_{10}$ in binary?

$$\frac{13}{2} = 6\ R\ \mathbf{1}$$

$$\frac{6}{2} = 3\ R\ \mathbf{0}$$

$$\frac{3}{2} = 1\ R\ \mathbf{1}$$

$$\frac{1}{2} = 0\ R\ \mathbf{1}$$

Now, the base $2$ representation comes from reading off the <u>remainders</u> from bottom to top!

$$13_{10} = \mathbf{1101_2}$$

## DECIMAL TO BINARY – ALGORITHM

Given a decimal number $n$, **repeat** the following until the number is $0$:

- Divide $n$ by $2$ and **prepend** the **remainder** of the division.
- Let the new number be $n$ divided by $2$, rounded down.
  (i.e. the **quotient** of the previous division)

This is called the remainder method. There are other ways to do the conversion.

## EXAMPLE

To convert $57_{10}$ in binary:

$$\frac{57}{2} = 28\ R\ \mathbf{1}$$

$$\frac{28}{2} = 14\ R\ \mathbf{0}$$

$$\frac{14}{2} = 7\ R\ \mathbf{0}$$

$$\frac{7}{2} = 3\ R\ \mathbf{1}$$

$$\frac{3}{2} = 1\ R\ \mathbf{1}$$

$$\frac{1}{2} = 0\ R\ \mathbf{1}$$

$\uparrow$

**Output:** 111001

## CHECK THE ANSWER!

- We think $57_{10} = 111001_2$

$$111001_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

$$= 32 + 16 + 8 + 1$$

$$= 57$$

✓

## ALGORITHM – BASE CONVERSION

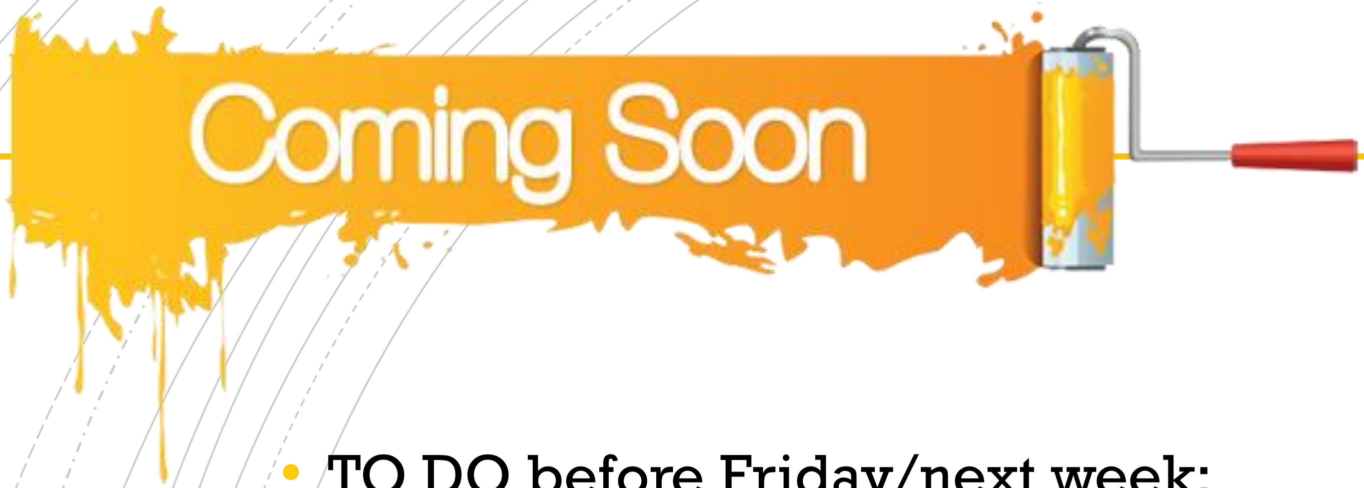
The technique just shown works in every base.

In general, given a base $b$ and a decimal number $n$, repeat the following until the number is $0$:

- Divide $n$ by $b$ and prepend the remainder of the division.
- Let the new number be $n$ divided by $b$, rounded down.

**ALGORITHM**
**Constructing Base $b$ Expansions**

```
**procedure** *BaseExpansion*$(n, b)$
$q := n$
$k := 0$
**While** $q \neq 0$
    $a_k := q \bmod b$
    $q := q \div b$
    $k := k + 1$
**return** $(a_{k-1}, \dots, a_1, a_0)$
```



- TO DO before Friday/next week:
  Install the JDK and Dr. Java on your laptop (or the machine you intend to work on)
- Read Appendix A and Chapter 1 from Think Java.
- On Friday:
  - Intro to programming languages and Java
  - Our first program: Hello World!
  - and more!