ETH zürich

# Session 01
# Integer Arithmetic

**Informatik I**
Lan Zhang
21.02.2025

# Schedule

1. Course Organization
2. Main Topics
   - Division and Modulo Operators
   - Number Systems
   - Expressions
3. In-Class Code Example

# General Info

Background:

4th Semester in ITET BSc

Programming experience:

HTML, CSS, PHP, Python, C++/C, JavaScript, SQL, MATLAB, HDL, Verilog, Bash, Rust

Interests:

Quantum Computing, Bioinformatics, Machine Learning

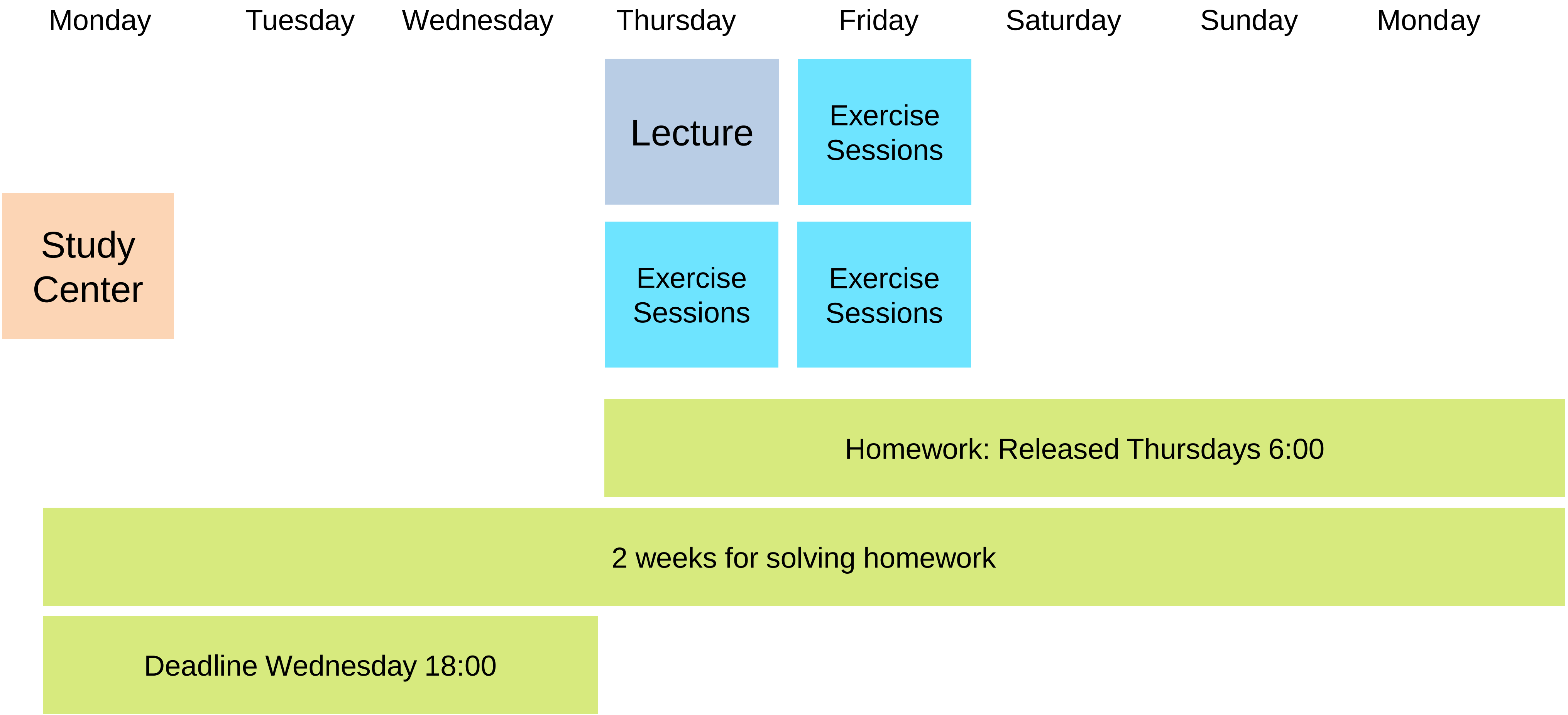
Expectations:

Participate in class

Practice and grasp concepts early on

Interrupt me and ask in case of ambiguities or doubts

# Weekly Schedule



Monday Tuesday Wednesday Thursday Friday Saturday Sunday Monday

Lecture

Exercise Sessions

Study Center

Exercise Sessions

Exercise Sessions

Homework: Released Thursdays 6:00

2 weeks for solving homework

Deadline Wednesday 18:00

# Weekly and Bonus exercises

- **Weekly exercises:**
  - Purpose: practice the new material
  - Released: Thursday at 6:00.
  - Deadline: A bit less than two weeks later (Wednesday at 18:00).
  - Allow earning experience points (XP)

- **Bonus exercises** (need around 2/3 experience points to unlock):
  - Purpose: combine knowledge from different topics
  - Allow earning max +0.25 towards the final grade (with 2/3 of bonus points)

# Weekly and Bonu

- **Weekly exercises**
  - Purpose: practice t
  - Released: Thursday at 6:00.
  - Deadline: A bit less
  - Allow earning expe
- **Bonus exercises (**
  - Purpose: combine knowledge from different topics
  - Allow earning max

Note:

- Only use constructs already introduced in the lecture and as specified in the task description
- Check the output of the autograder to view passed testcases and compiler warnings (0 pts if not corrected)

Study Center in GLC E29.2

- chance to ask for individual help regarding the course
- Time: Monday 12:15-14:00, starting in the second week of the semester

Exam

- 20~25p Theory Questions
- 5 Coding Examples with 10~15p each

# Integers

- Storage as binary numbers, bitsize dependent on OS: in C++ -> 32 bits
- Unsigned Integers $\mathbb{Z}$: $[0,2^{32}-1]$ -> $2^{32}$ -1 = uint_max = -1, $2^{32}$ = 0
- Signed Integers $\mathbb{N}$: $[-2^{31},2^{31}-1]$ , 1 bit for sign
  - 2’s complement: 1. Convert the absolute value of x to binary. 2. Flip bits. 3. Add 1.

- Under-/Overflow when number exceeds min./max. value representable by data type
  - Overflow Detection of a+b: a < (int_max-b)

```
#include <limits>

std::numeric_limits<T>::min()/max()
```

# Division and Modulo Operators

- Integer division ignores digits behind the decimal point: (always rounds towards 0)

```
7 / 3 == 2
15 / 4 == 3
16 / 4 == 4
```

- Modulo division gets the remainder:

```
7 % 3 == 1
15 % 4 == 3
16 % 4 == 0
```

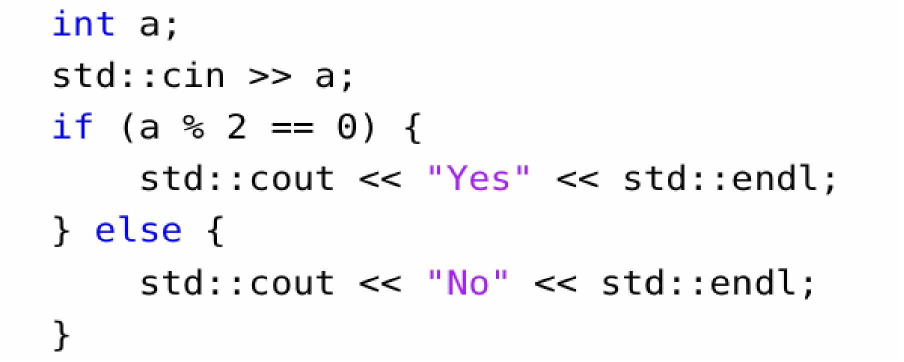
- The original number can be obtained like this:

```
(a/b)*b+(a%b)==a
```

**Note**: The modulo operator is not suitable for non-integer numbers!

# Division and Modulo Operators

- What question is answered by the output of the following code?



```
int a;
std::cin >> a;
if (a % 2 == 0) {
    std::cout << "Yes" << std::endl;
} else {
    std::cout << "No" << std::endl;
}
```

Check divisibility: `a%b == 0 -> b|a`

Negative Modulo:

- takes sign of left term e.g. `(-7)%5= 2, 9%(-4)=1`
- add m

Rules:

- `(a+b)%m = ((a%m)+(b%m))%m`
- `(a*b)%m = ((a%m)*(b%m))%m`

# Number Systems

Method of expressing numbers

- decimal, binary 0b, hexadecimal 0x, octal 0

Base n system

- digits in [0, n-1]
- represented as powers of n

```
convert_base_from_decimal
  unsigned int result = 0;
  unsigned int basetenposition = 1;
  while (n != 0) {
    result += basetenposition * (n % b);
    n = n / b;
    basetenposition *= 10;
  }
```

# Expressions

Precedence and Associativity

- Higher precedence operators are evaluated first
- If same precedence, then evaluate according to associativity
- Order of operations:
  1. Binary arithmetic operators ( +, −, ... )
  2. Relational operators (<, >, ... )
  3. Binary logical operators ( &&, ... )

L- and R-values

- semantic properties of expressions
- L-value: refers to an object, has an address in memory, can change its value.
- R-value: literals, cannot change its value. L-value can be used as R-value but not the other way around.

Attention: (a+b)*(a++), evaluation order depends on the compiler! Avoid!

# In-Class Code Example

Last 3 Digits: Write a C++ program which, for an integer $x \in Z_{\geq 1000}$, outputs the last 3 digits

In general: For $n \in N$ and $x \in Z_{\geq 10^n}$ , $(x\%10^n - x\%10^{n-1})/10^{n-1}$ corresponds to the nth digit of x

Question: How could we change the program from the previous task so that it outputs the last 3 bits?

Solution: Change all 10 into 2. (i.e. into the base of the system into which we are converting)

[code]expert basics:

- code snippets for auto-complete https://github.com/ajaxorg/ace/blob/master/src/snippets/c_cpp.snippets.js
- use pythontutor for interactive code execution https://pythontutor.com/cpp.html#mode=edit