

# **CSE215**

## **Foundations of Computer Science**

**Instructor: Zhoulai Fu**

**State University of New York, Korea**

Course Website:  
[https://github.com/zhoulai fu/23\\_cse215\\_fall](https://github.com/zhoulai fu/23_cse215_fall)

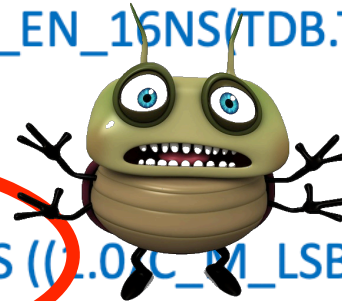
# Why Studying Computer Science?

# French Ariane 5 Rocket, 1996



# Ada code for Ariane 5 Rocket

```
if L_M_BV_32 > 32767 then
  P_M_DERIVE(T_ALG.E_BV) := 16#7FFF#;
elsif L_M_BV_32 < -32768 then
  P_M_DERIVE(T_ALG.E_BV) := 16#8000#;
else
  P_M_DERIVE(T_ALG.E_BV) := UC_16S_EN_16NS/TDB.T_ENTIER_16S(L_M_BV_32));
end if;
P_M_DERIVE(T_ALG.E_BH) :=
  UC_16S_EN_16NS (TDB.T_ENTIER_16S ((1.0/C_M_LSB_BH)*G_M_INFO_DERIVE(T_ALG.E_BH))));
```



## \$7 billion Software Disaster

### Comparison:

SUNY Korea was awarded \$0.05 billion for 10 years under an MKE grant  
(Source: <https://sunyk.cs.stonybrook.edu/>)

From 2018 to 2020, South Korea GDP dropped \$94 billion;  
(Source: World bank)

- How to make reliable software?
- How to make efficient software?
- How to make energy-friendly software?
- We need to understand **deeply** how code works

- We need to understand **deeply** how code works
- Quiz: If Precondition holds, will Postcondition holds after executing this piece of code?

```
Precondition:  $x \geq 0$ ;
```

```
z = 0;
```

```
if (x != 0)
```

```
    z = x;
```

```
} else {
```

```
    z = z+1
```

```
}
```

```
Postcondition:  $z > 0$ ;
```



- We need to understand **deeply** how code works

```
Precondition:  $x \geq 0$ ;  
  
z = 0;  
                                     {  $x \geq 0 \ \&\& \ z = 0$  }  
if (x != 0) {  
    z = x;  
                                     {  $x > 0 \ \&\& \ z = 0$  }  
                                     {  $x > 0 \ \&\& \ z = x$  }  
} else {  
                                     {  $x = 0 \ \&\& \ z = 0$  }  
    z = z+1;  
                                     {  $x = 0 \ \&\& \ z = 1$  }  
}  
Postcondition:  $z > 0$ ;
```

- Yes. We can **prove** it!

Precondition:  $x \geq 0$ ;

$z = 0$ ;

$\{ x \geq 0 \ \&\& \ z = 0 \}$

if ( $x \neq 0$ ) {

$\{ x > 0 \ \&\& \ z = 0 \}$

$z = x$ ;

$\{ x > 0 \ \&\& \ z = x \}$

} else {

$\{ x = 0 \ \&\& \ z = 0 \}$

$z = z + 1$

$\{ x = 0 \ \&\& \ z = 1 \}$

}

Postcondition:  $z > 0$ ;



**Propositional  
Logic**

**Predicate  
Logic**

**Proof**

**CSE215**

**Sequences**

**Sets**

**Functions**

**Relations**

# Expected Learning Outcomes

- An ability to check if a mathematical argument is valid
- An ability to verify the correctness of proofs of some existing theorems and prove some new theorems
- An ability to use the mathematical concepts of sequences, functions, relations, and sets in solving computing problems

# Logistic matters

- Team
- Textbook
- Schedule
- Homework
- Exams and grading
- Ask for help

# Meet the Instructor

- Teaching CSE215 and CSE216
- Data & Intelligent Computing Lab (C404)
- Previous Work: France, US, Denmark and Korea
- Education: École Polytechnique, France
- Personal: Happily married; like dreaming and playing with my child; no special hobbies or talents.

# TA

- Jun Hyeong Park <[junhyeong.park@stonybrook.edu](mailto:junhyeong.park@stonybrook.edu)>

# Team

**You**

**TA**

**Instructor ChatGPT**

**Lectures**

**Office hours**

**Not do  
homework**

**Office hours**

**Lectures**

**Homework**

**Grading**

**Answer  
questions**

**Answer  
questions**

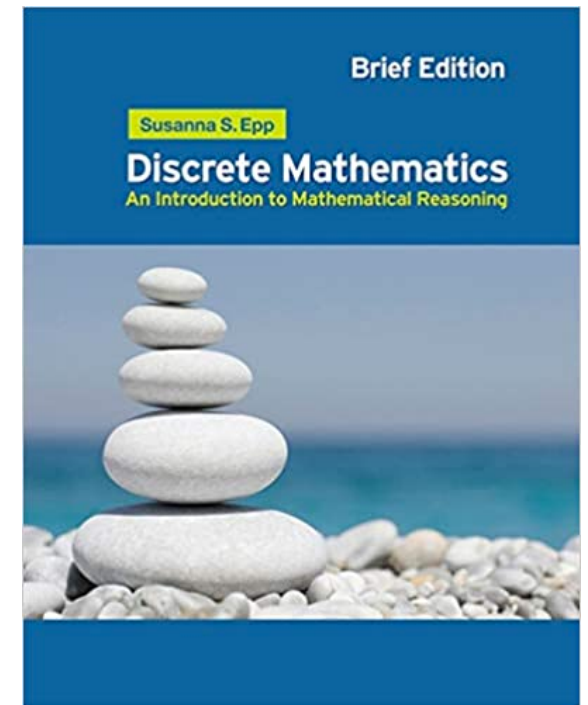
**Answer  
questions**

**Ask questions**

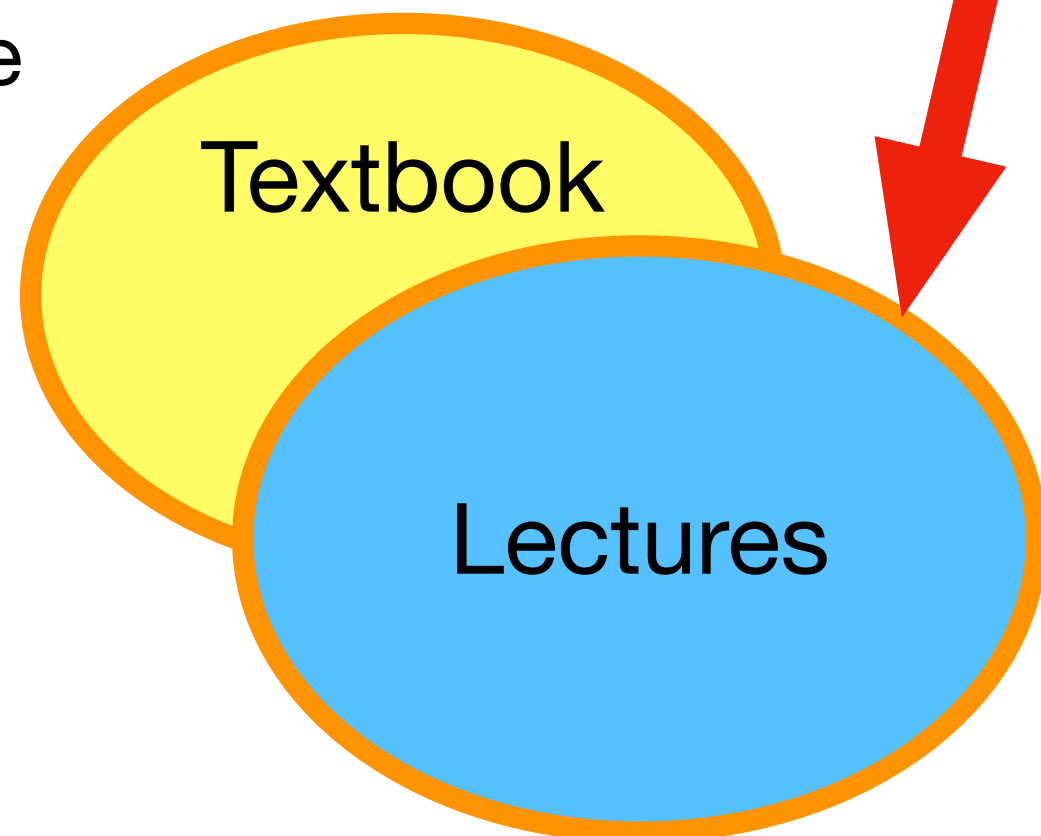


# Textbook

- Our course relates to Chapters 2-7
- Very helpful, but optional
- Suggestion: Skim the related chapter before the lecture; read deeper after the lecture
- Textbook may not cover everything in the exams; lectures do



Exam



# Schedule

- Lectures: TU, TH 12:30 - 13:50, at B207
- Recitation: TU 15:30 - 16:25, at B207
- Homework is announced on Thursday, and due time is next Thursday 23:59 (included)
- Office hours: TU 14:00 - 15:00, TH 15:30 - 16:30 at B424
- TA Office hours: Mo 18:00-22:00, We 16:00 - 20:30
- Final: Dec 12 Tuesday 12:30 - 15:00 pm at B207

# Numerical Grading

- Homework: 30%
- Midterm1: 20%
- Midterm2: 20%
- Final: 30%
- Students with regular participation or constructive feedback get 0.5% or 1% bonus

# Letter Grade

- Absolute grading if median  $\geq 90$
- Relative grading otherwise
- Algorithm guarantees 50% A/A-
- Details: [https://github.com/zhoulaiifu/23\\_cse215\\_fall/blob/master/README.md#grading-letter-scores](https://github.com/zhoulaiifu/23_cse215_fall/blob/master/README.md#grading-letter-scores)

```
def get_letter_grade(score, c):  
    boundaries = {  
        'A': 0.7*c + 30,  
        'A-': c,  
        'B+': c - 4,  
        'B': c - 7,  
        'B-': c - 10,  
        'C+': c - 14,  
        'C': c - 17,  
        'C-': c - 20,  
        'D+': c - 24,  
        'D': c - 27,  
        'D-': c - 30,  
    }  
  
    for grade, boundary in boundaries.items():  
        if score > boundary:  
            return grade  
    return 'F'
```

# Recipe for Success in CSE215

- Attend lectures
- Ask questions
- **Do homework (VITAL)**



# Quiz

- What is a super important thing to succeed in the class?
- Is ChatGPT allowed?
- Homework due time?
- How to ask for help?
- Where to find official course info?

**Questions so far?**

# Course overview

# A personal story

- Once upon a time, I worked for a project involving financial calculation
- I needed to sum up a number of floating-point values like
  - $0.1 + 0.2 + 0.3 + 0.7 + 0.9 + 1.2 + 3.5 \dots$
- There were billion of numbers like this, so performance was a key for the project's success
- We decided to use the state-of-the-art multi-core, parallel computing
- Parallel computing works like a divide-and-conquer:
  - $(0.1 + 0.2) + (0.3 + 0.7) + (0.9 + 1.2 + 3.5) + \dots$
- Now, let us think why it looks reasonable to use parallel computing for this task??
- The reason is associative law.  $(a + b) + c = a + (b + c)$

# A problem

We get different results for each round, if we put parentheses differently each time.

- $(0.1 + 0.2) + (0.3 + 0.7) + (0.9 + 1.2 + 3.5) + \dots$
- becomes different from
- $(0.1 + 0.2 + 0.3) + (0.7 + 0.9) + (1.2 + 3.5) + \dots$

Demo: <https://www.pythonanywhere.com/try-ipython/>

# Why?

- We made this assumption:
  - for any numbers  $a, b, c$ ,  $(a + b) + c = a + (b + c)$
- This is a statement that can be assigned with true or false value, we call it a **proposition**
- The inner part has variables, and can be denoted as a statement with parameters  $(a, b, c)$ . We call it a **predicate**.
- Many CS work involves determining if a proposition is true or false. To show the truth is called **to prove**.
- The reason for the problem is that the proposition above is false.



# Summary for the story

The whole is called a proposition,  
to which we can assign a truth value

$$\forall a, b, c \in I, (a + b) + c = a + (b + c)$$



Predicate

quantifier

variables

set

# Summary

- The ultimate goal of this course is to learn fundamentals for understanding why our digital world works or fails.
- We will study logic (propositions, and predicates), proof, and math structures like sets as a language to reason about computer science
- Our next classes will be about propositions.