COSE212: Programming Languages

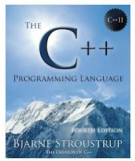
Lecture 18 — Course Review

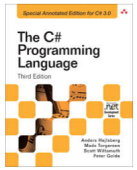
Hakjoo Oh
2022 Fall

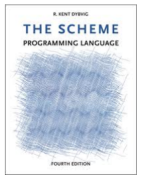
About This Course (from Lecture 0)

This course is *not* about

- to learn particular programming languages








- to improve your "programming skills" (e.g., tools, libraries, etc)

Instead, in this course you will learn

- fundamental principles of modern programming languages
- how programming systems are designed and implemented
- thinking formally and rigorously

To succeed in this course, you must

- have basic programming skills
- be familiar with at least two PLs (e.g., C, Java)
- have taken Theory of Computation, Discrete Math, etc
- be prepared to learn new things

Design and Implementation of Programming Languages (from Lecture 0)

We will learn programming language concepts by designing and implementing our own programming language system.

- We will define a programming language. For example, "factorial" is written in our language as follows:

```
let x = read in
letrec fact(n) =
  if iszero n then 1
  else ((fact (n-1)) * n)
in (fact x)
```

- We will design and implement an interpreter for the language:

$$\text{Program} \rightarrow \boxed{\text{Interpreter}} \rightarrow \text{Result}$$

- We will design and implement a type checker for the language:

$$\text{Program} \rightarrow \boxed{\text{Type Checker}} \rightarrow \text{Safe/Unsafe}$$

Checklist

Have you pick up the following ideas from this course?

- Designing programming languages (i.e., syntax and semantics)
- Implementing programming languages (i.e., interpreters)
- Detecting runtime errors at compile-time (i.e., type system)

Applications of Programming Language Foundations

A good understanding of programming language foundations is essential for

- Software engineering
- Software security
- Software analysis
- ...

Application (1): Software Bug/Vulnerability Detection

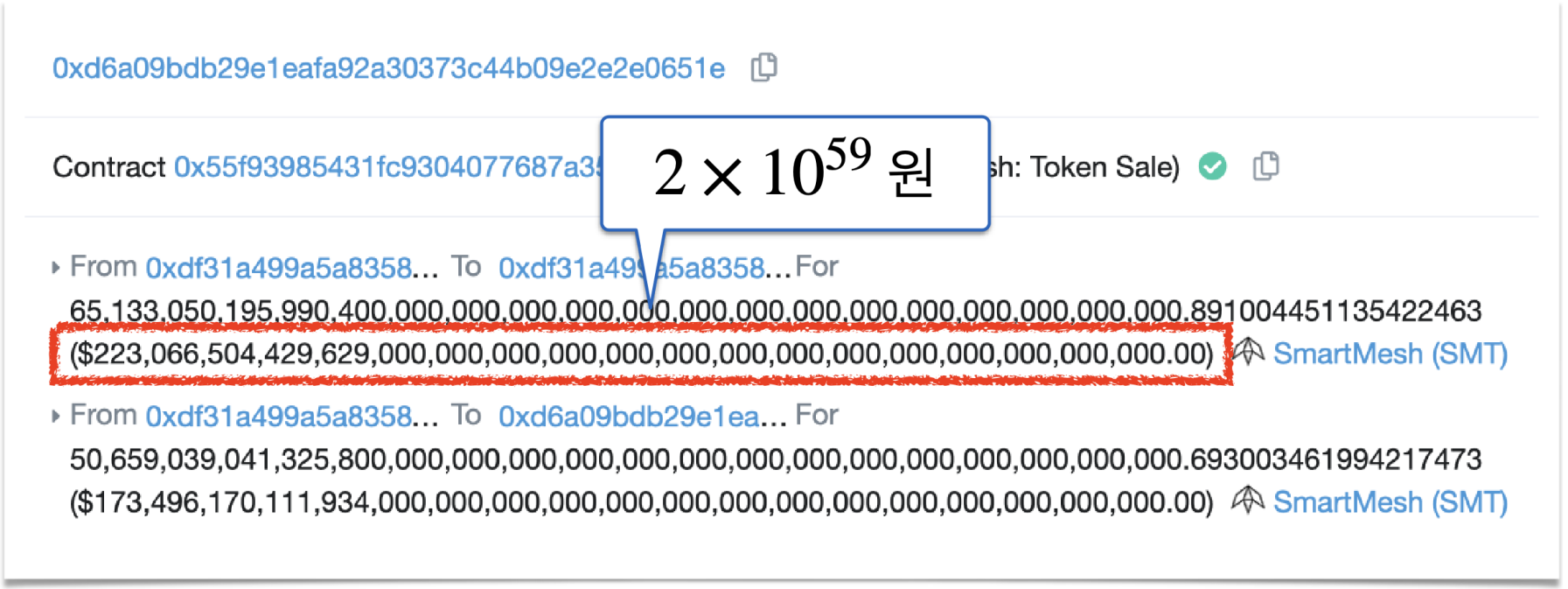
스마트 컨트랙트는 안전성 검증이 필수

- 한번 배포되면 코드 수정 불가능
- 공격에 성공하면 막대한 금전적 피해 발생

Application (1): Software Bug/Vulnerability Detection

SmartMesh 사례 (2018)

- 정수 오버플로우 취약점을 이용하여 천문학적 금액의 토큰을 생성 (CVE-2018-10376)



https://etherscan.io/tx/0x1abab4c8db9a30e703114528e31dee129a3a758f7f8abc3b6494aad3d304e43f

SmartMesh 사례 (2018)

- 정수 오버플로우 (integer overflow) 취약점
- 방어적으로 코드를 작성했음에도 문제가 된 경우

```
1   function transferProxy (address from, address to, uint
        value, uint fee) public returns
2     if (balance[from] < fee + value)
3       revert();
4     if (balance[to] + value < balance[to] ||
5         balance[msg.sender] + fee < balance[msg.sender])
6       revert();
7     balance[to] += value;
8     balance[msg.sender] += fee;
9     balance[from] -= value + fee;
10    return true;
11  }
```

보내는 사람의 잔고가 충분한지 체크

오버플로우 체크

송금

(실질적) 오버플로우/언더플로우 발생하지 않음

SmartMesh 사례 (2018)

```
balance[from] = balance[to] = balance[msg.sender] = 0
value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

```
1   function transferProxy (address from, address to, uint
          value, uint fee) public returns (bool) {
2       if (balance[from] < fee + value)
3           revert();
4       if (balance[to] + value < balance[to] ||
5           balance[msg.sender] + fee < balance[msg.sender])
6           revert();
7       balance[to] += value;
8       balance[msg.sender] += fee;
9       balance[from] -= value + fee;
10      return true;
11  }
```

false

0!

false

8fffff···ff

700···00

0!

Application (1): Software Bug/Vulnerability Detection

기존 취약점 검출기와 성능 비교

정확도: 99.5%
검출률: 100%

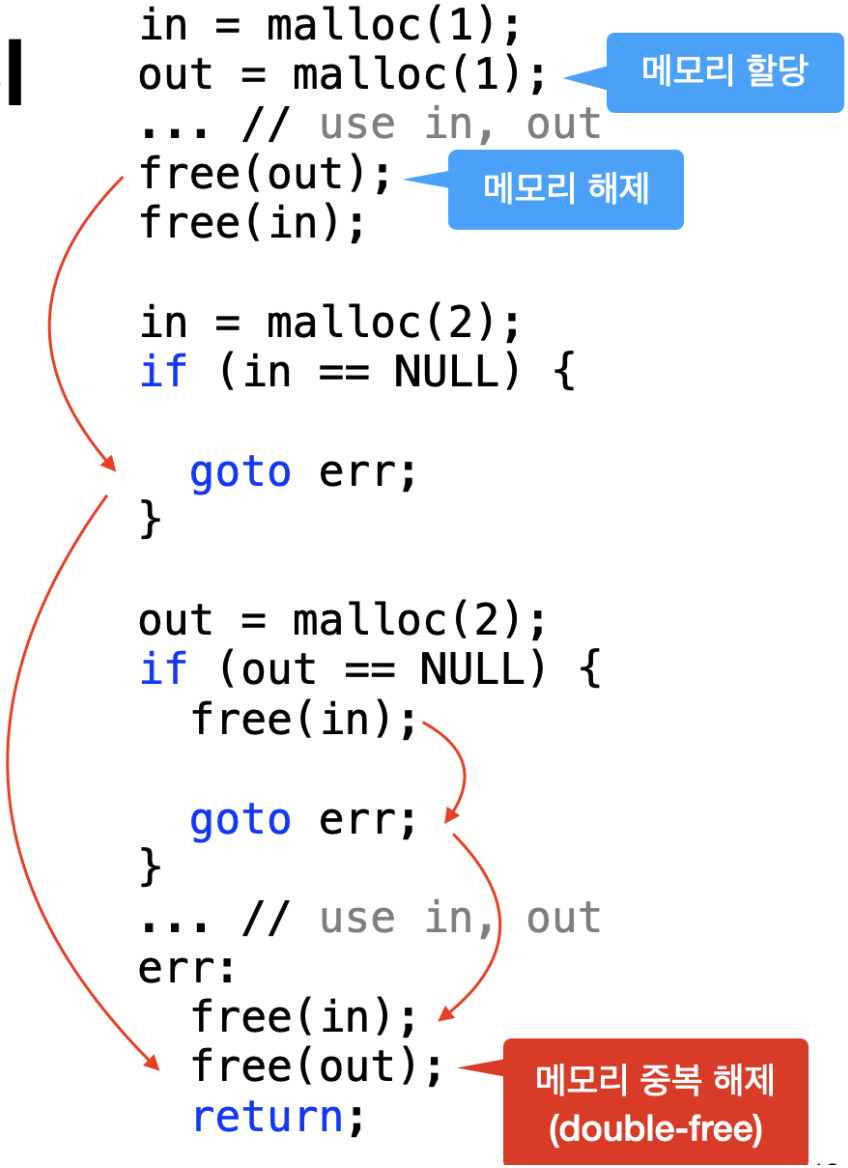
정확도: < 94.6%
검출률: < 70.7%

No.	CVE ID	Name	LOC	#Q	VeriSmart #Alarm	VeriSmart #FP	VeriSmart CVE	Osiris [7] #Alarm	Osiris [7] #FP	Osiris [7] CVE	Oyente [9, 26] #Alarm	Oyente [9, 26] #FP	Oyente [9, 26] CVE	Mythril [8] #Alarm	Mythril [8] #FP	Mythril [8] CVE	MantiCore [10] #Alarm	MantiCore [10] #FP	MantiCore [10] CVE
#1	2018-10299	BEC	299	6	2	0	✓	0	0	✗	1	0	△	2	0	✓	0	0	✗
#2	2018-10376	SMT	294	22	13	0	✓	1	0	✓	2	0	✗	1	0	✗	timeout (> 3 days)		
#3	2018-10468	UET	146	27	14	0	✓	9	0	✗	8	0	✓	5	0	✓	0	0	✗
#4	2018-10706	SCA	404	48	33	0	✓	9	0	✗	4	0	△	2	0	✗	internal error		
#5	2018-11239	HXG	102	11	7	0	✓	6	0	✓	2	0	✗	3	0	✓	2	0	✓
#6	2018-11411	DimonCoin	126	15	7	0	✓	5	0	✗	5	0	✓	5	0	✓	3	0	✓
#7	2018-11429	ATL						3	0	✓	2	0	∠						
#8	2018-11446	GRX						8	2	✗	12	4	✗						
#9	2018-11561	EET						4	0	✓	2	0	∠						
#10	2018-11687	BTC						2	0	✓	2	0	∠						
#11	2018-12070	SEC						6	0	✓	4	0	✗						
#12	2018-12230	RMC						3	0	✓	5	0	✓						
#13	2018-13113	ETT						4	2	N/A	2	2	N/						
#14	2018-13126	Mox						0	0	✗	0	0	✗						
#15	2018-13127	DSP						3	0	✓	3	0	∠						
#16	2018-13128	ETY						3	0	✓	3	0	∠						
#17	2018-13129	SPX						5	0	✓	3	0	∠						
#18	2018-13131	SpadePreSale	312	4	3	0	✓	0	0	✗	0	0	✗	0	0	✗	internal error		

			VeriSmart #Alarm	VeriSmart #FP	VeriSmart CVE	Osiris [43] #Alarm	Osiris [43] #FP	Osiris [43] CVE	Oyente [9, 34] #Alarm	Oyente [9, 34] #FP	Oyente [9, 34] CVE	Mythril [7] #Alarm	Mythril [7] #FP	Mythril [7] CVE	MantiCore [2] #Alarm	MantiCore [2] #FP	MantiCore [2] CVE
Total	12493	976	492	2	✓: 58 △: 0 ✗: 0	240	13	✓:41 △: 0 ✗ :17	171	14	✓:20 △:15 ✗ :23	94	10	✓:10 △: 1 ✗ :46	14	0	✓: 2 △: 0 ✗ :42

Application (2): Software Error Repair

사례 1: Linux Kernel



Application (2): Software Error Repair

사례 1: Linux Kernel

수동 디버깅의 문제 1:
오류가 제거되었는지 확신하기 어려움

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {
  out = NULL;
  goto err;
}

out = malloc(2);
if (out == NULL) {
  free(in);
  in = NULL;
  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

Application (2): Software Error Repair

사례 1: Linux Kernel

memory leak

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {
  out = NULL;
  goto err;
}
free(out);
out = malloc(2);
if (out == NULL) {
  free(in);
  in = NULL;
  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

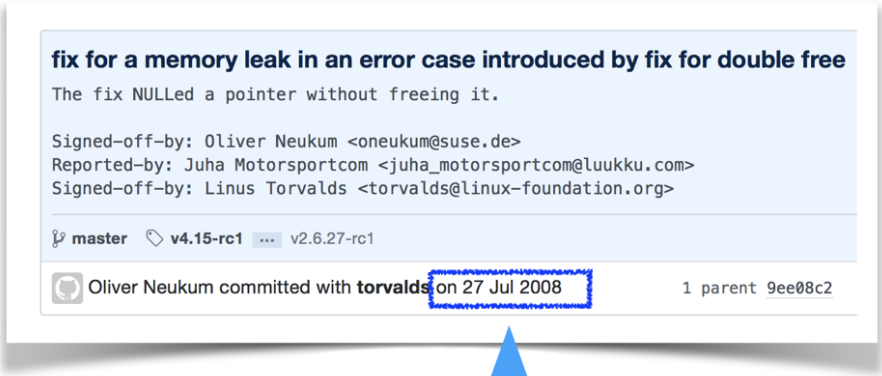
수동 디버깅의 문제 2:
오류 수정 과정에서 새로운 오류가 발생

9개월 후에 다시 오류 수정을 시도

Application (2): Software Error Repair

사례 1: Linux Kernel

수동 디버깅의 문제 3:
오류는 제거했지만 코드 품질이 떨어짐



오류 발견에서 수정까지 총 10개월 소요

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);
out = NULL;
in = malloc(2);
if (in == NULL) {
  out = NULL;
  goto err;
}
free(out);
out = malloc(2);
if (out == NULL) {
  free(in);
  in = NULL;
  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

Application (2): Software Error Repair

ICSE 2020

SAVER: 메모리 오류 자동 수정기

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {

  goto err;
}

out = malloc(2);
if (out == NULL) {
  free(in);

  goto err;
}
... // use in, out
err:
  free(in); // double-free
  free(out);// double-free
  return;
```

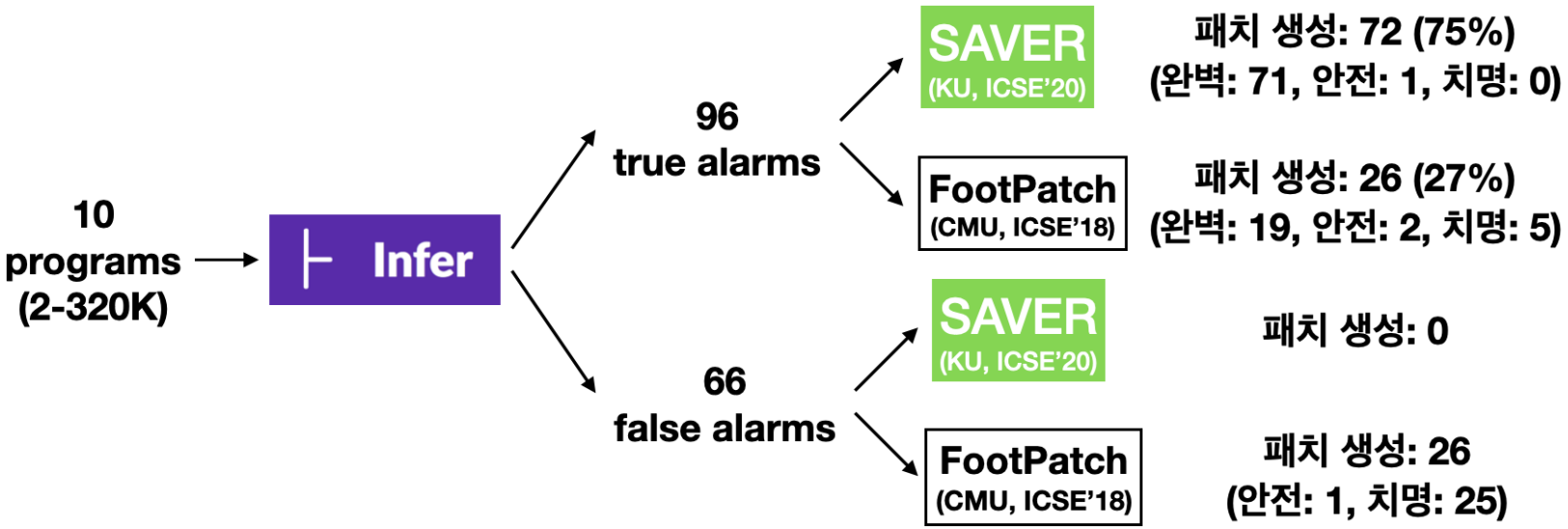
✓개발생산성↑
✓SW품질↑
✓안전성 보장

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {

  goto err;
}
free(out);
out = malloc(2);
if (out == NULL) {
  free(in);

  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

Application (2): Software Error Repair

SAVER 성능

		Infer				SAVER						FootPatch [60]						
Program	kLoC	#T	#F	Pre(s)	Fix(s)	G_T	$✓_T$	$\triangle_T$	$✗_T$	G_F	$✗_F$	Fix(s)	G_T	$✓_T$	$\triangle_T$	$✗_T$	G_F	$✗_F$
rappel (ad8efd7)	2.2	1	0	2.2	0.0	1	1	0	0	0	0	8.9	1	1	0	0	0	0
flex (d3de49f)	22.3	3	4	26.3	2.5	0	0	0	0	0	0	51.0	0	0	0	0	1	1
WavPack (22977b2)	31.2	1	2	44.6	22.1	0	0	0	0	0	0	67.9	0	0	0	0	2	2
Swoole (a4256e4)	43.0	15	3	88.5	10.1	11	11	0	0	0	0	392.5	9	7	0	2	1	1
lxc (72cc48f)	49.9	3	5	230.6	5.8	3	3	0	0	0	0	179.6	0	0	0	0	1	1
p11-kit (ead7ara)	62.9	33	9	646.2	288.8	24	24	0	0	0	0	566.4	8	7	1	0	2	2
x264 (d4099dd)	73.2	10	0	144.3	9.9	10	10	0	0	0	0	426.9	2	2	0	0	0	0
recutils-1.8	92.0	10	11	144.1	44.4	8	8	0	0	0	0	662.2	3	2	1	0	0	0
inetutils-1.9.4	116.9	4	5	44.8	2.5	4	4	0	0	0	0	182.1	0	0	0	0	0	0
snort-2.9.13	320.8	16	27	2372.0	216.0	11	10	1	0	0	0	4636.4	3	0	0	3	19	18
Total	814.4	96	66	3743.6	602.1	72	71	1	0	0	0	7173.9	26	19	2	5	26	25



Application (2): Software Error Repair

Application to Intelligent Tutoring System

- 오류 수정 기술을 함수형 프로그래밍 교육에 적용
- 현재 코딩 교육 자동 도구들의 한계: 개인화된 피드백 제공 못함

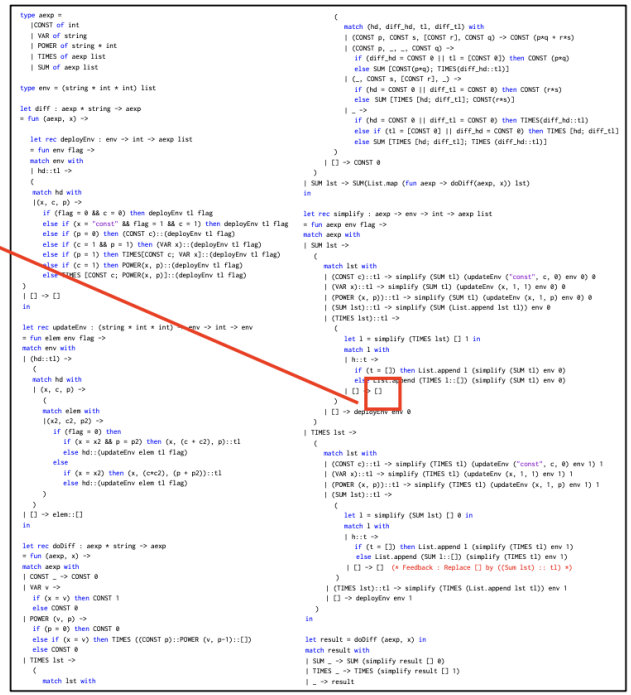
CodeOnWeb

oncoder

FixML-generated feedback: ((Sum lst)::tl)

```
let rec diff : aexp * string -> aexp
= fun (e, x) ->
   match e with
   | Const n -> Const 0
   | Var a -> if (a <> x) then Const 0 else Const 1
   | Power (a, n) -> if (a <> x) then Const 0 else Times [Const n; Power (a, n-1)]
   | Times l ->
      begin
      match l with
      | [] -> Const 0
      | hd::tl -> Sum [Times ((diff (hd, x))::tl); Times [hd; diff (Times tl, x)]]
      end
   | Sum l -> Sum (List.map (fun e -> diff (e,x)) l)
```

제공된 솔루션



학생 제출 답안

Application (3): Automatic Software Synthesis

```python
# Write a python function to toggle all even bits of a given number.
# Your code should pass these tests:
#
# assert even_bit_toggle_number(10) == 0
# assert even_bit_toggle_number(20) == 30
# assert even_bit_toggle_number(30) == 20
def even_bit_toggle_number(n):
    count = 0
    res = 0
    temp = n
    while temp > 0:
        if count % 2 == 1:
            res |= 1 << count
        count += 1
        temp >>= 1
    return n ^ res
```

References

- Sunbeom So, Myungho Lee, Jisu Park, Heejo Lee, and Hakjoo Oh.
  VeriSmart: A Highly Precise Safety Verifier for Ethereum Smart Contracts.
  S&P 2020: 41st IEEE Symposium on Security and Privacy
- Dowon Song, Woosuk Lee, and Hakjoo Oh.
  Context-Aware and Data-Driven Feedback Generation for Programming Assignments.
  ESEC/FSE 2021: ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering
- Junhee Lee, Seongjoon Hong, and Hakjoo Oh.
  NPEX: Repairing Java Null Pointer Exceptions without Tests.
  ICSE 2022: International Conference on Software Engineering

한 학기 수고 많았습니다!