PIC 10A 1A

TA: Bumsu Kim

UCLA

Today…

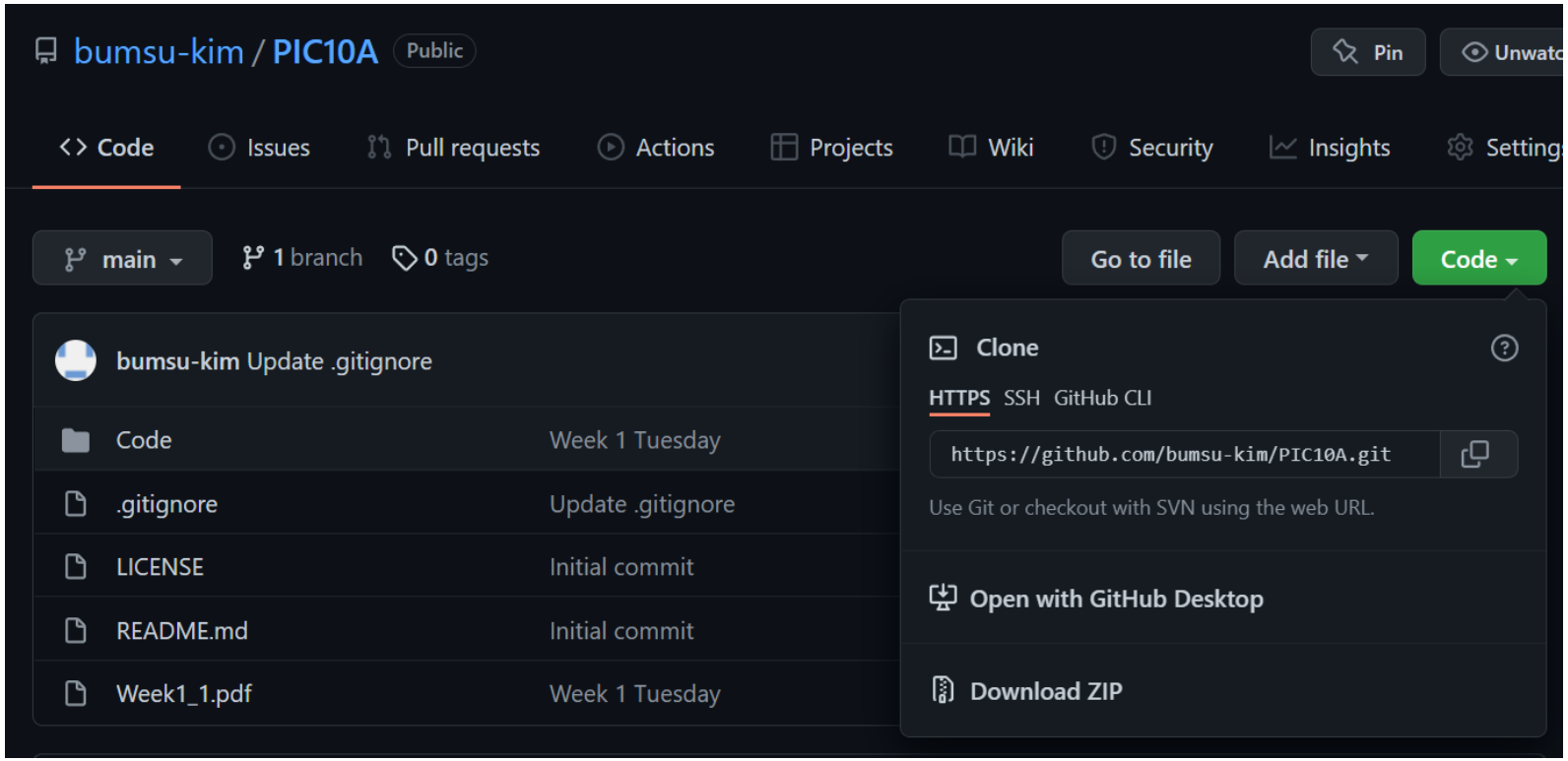
- Github and Google Form
- Variables, Bits and Types
- The type `int` and Arithmetic Operations

UCLA

Github Repo

- Go to: https://github.com/bumsu-kim/PIC10A

- Discussion slides and supplementary materials (e.g. code) will be uploaded there



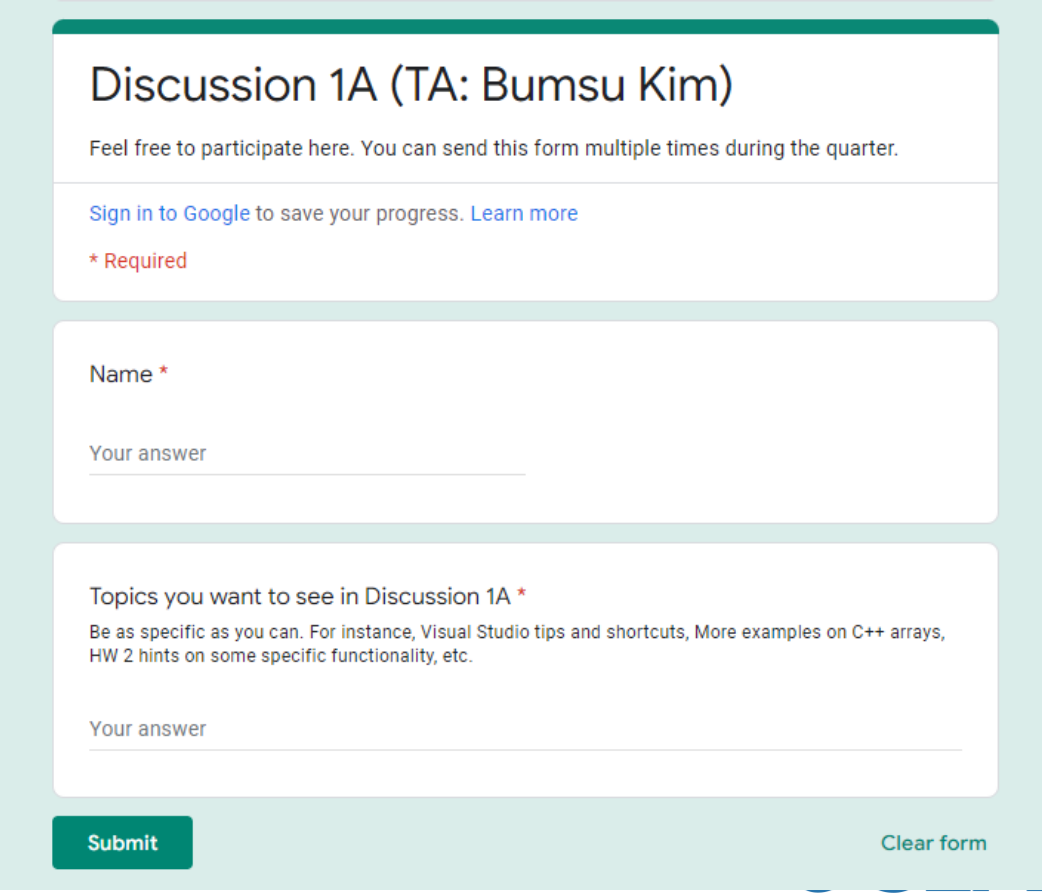
3 options to download files

1. Use Git (may be difficult)
2. Use GitHub Desktop (easy to sync, once set up)
3. Download ZIP (easy but requires downloading the whole repo every time when it is updated)

Google Form

- Go to: BruinLearn – Module “Discussion 1A”

- Let me know what you want in Discussion. Please be specific. For example, you can write:
  - Visual Studio tips and shortcuts
  - More examples on C++ int arithmetics
  - Live coding exercise for functions
  - How to find a file that I just created
  - Hints for HW0 Exercise 2



Variables

- C++ is case sensitive. For example,

```
#include <iostream>
using namespace std;

int Main() {
        cout << "Hello, World!" << endl;
        return 0;
}
```

will not compile

UCLA

Variables

- C++ is case sensitive. For example,

```
#include <iostream>
using namespace std;

int Main() {
        cout << "Hello, World!" << endl;
        return 0;
}
```

will not compile. Neither will

```
#include <iostream>
using namespace std;

int main() {
        int num = 2;
        cout << Num << endl;;
        return 0;
}
```

UCLA

Variables

- What is a variable?

```
#include <iostream>
using namespace std;

int main() {
    int my_number;
    my_number = 5;
    cout << "The content of my_num is ";
    cout << my_number << "." << endl;
    return 0;
}
```

- The output is:

```
C:\Windows\system32\cmd.exe
The content of my_num is 5.
Press any key to continue . . .
```

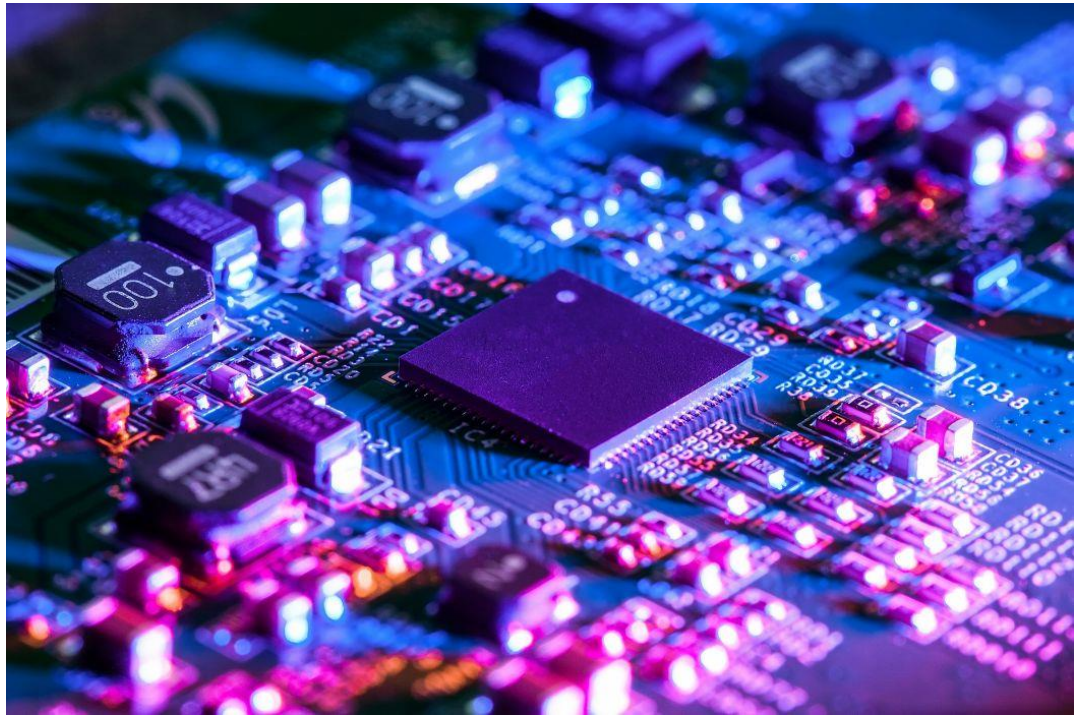
UCLA

Types

- Computers use **bits** to store and process data
  - 8 bits = 1 byte (usually the smallest unit for this course)

- Bit
  - Binary digit
  - 0 or 1
- Byte
  - 8 bits

On (1) / Off (0) (electric signals)



Types

- Computers use **bit**s to store and process data
  - 8 bits = 1 byte (usually the smallest unit for this course)

- Bit
  - Binary digit
  - 0 or 1
- Byte
  - 8 bits

On (1) / Off (0) (electric signals)

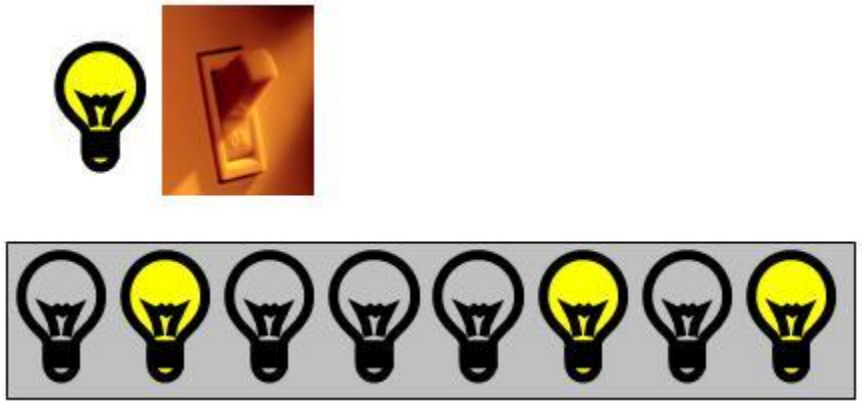
- Naturally corresponds to binary numbers
  - 01000101 (base 2)
- Floating point numbers, strings, and even graphics are essentially represented as combinations of bits

UCLA

Types

- Everything (a floating point number, a string, and even a graphical component) is essentially represented as combinations of bits in your computer

- Bit
  - Binary digit
  - 0 or 1
- Byte
  - 8 bits



On (1) / Off (0)

- So [01000101…] might mean an integer number, a floating point number, a character (like ‘a’), or even a part of an image or video.

→ You need to specify a “**type**” of the data so that the machine can interpret it as you intend

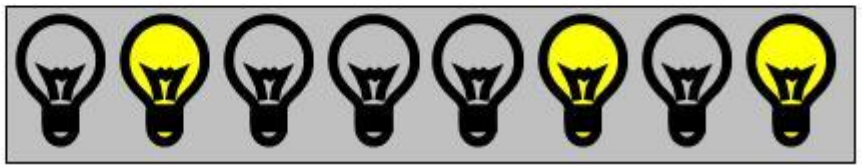
UCLA

Integral types (will be revisited next week)

- Integral type: uses base 2 representation of integers (binary numbers)
  - 8, 16, 32 or 64 **bit**s
  - = 1, 2, 4 or 8 **byte**s
  - ex) 0100 1111 0101 0000 (base 2)
  - $= 2^{14} + 2^{11} + 2^{10} + 2^9 + 2^8 + 2^6 + 2^4 = 20304$

- Q) How many different states can be represented by 8 bits?

- Bit
  - Binary digit
  - 0 or 1
- Byte
  - 8 bits



On (1) / Off (0)

UCLA

Integral types (will be revisited next week)

- Integral type: uses base 2 representation of integers (binary numbers)
  - 8, 16, 32 or 64 **bit**s
  - = 1, 2, 4 or 8 **byte**s
  - ex) 0100 1111 0101 0000 (base 2)
  - $= 2^{14} + 2^{11} + 2^{10} + 2^9 + 2^8 + 2^6 + 2^4 = 20304$

- If you use 16 bits (= 2bytes), you can represent total 2^{16} different numbers
  - If you start from 0, then from 0 to $2^{16} - 1$
  - If you want to include negative numbers, with 0 at the middle, then -2^{15} to $2^{15} - 1$

- The type `int` and `unsigned int` usually have 4 bytes (about 4 billion states)
- For larger/smaller numbers, use `(unsigned) long long int`

UCLA

When poll is active, respond at **pollev.com/bumsukim297**

Text **BUMSUKIM297** to **37607** once to join

What is the largest unsigned integer value that can be expressed with 4 binary bits of information?

3

4

7

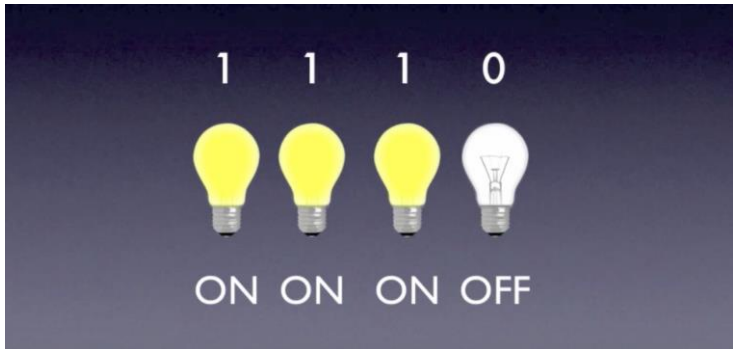
8

15

16

Powered by Poll Everywhere

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**



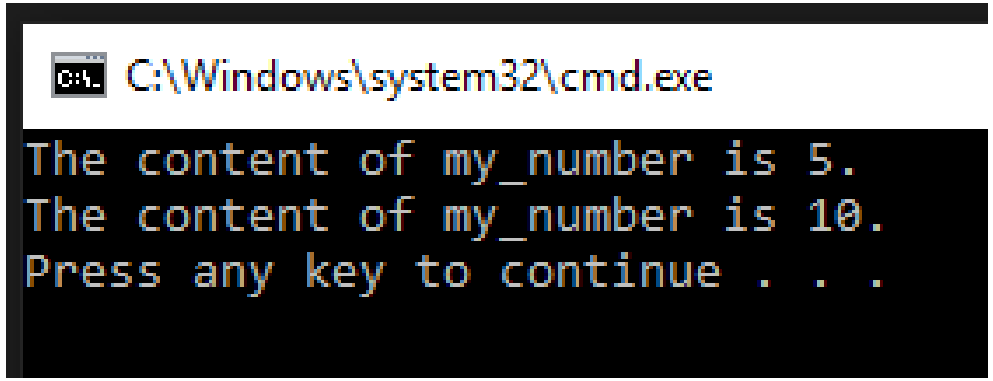
ANS: 15. We start counting from 0!

Variables

```
#include <iostream>
using namespace std;

int main() {
    int my_number;
    my_number = 5;
    cout << "The content of my_number is " << my_number << "." << endl;
    my_number = 10;
    cout << "The content of my_number is " << my_number << "." << endl;
    return 0;
}
```

- What is the output?



UCLA

Variables – Declaration

- Variable declaration/definition:

```
int my_number;
```

  - This introduces `my_number` as a *variable*
  - Its *type* is `int`, short for integer
  - A variable is a place to store information
  - In this case `my_number` is a place to store an integer

UCLA

Variables – Declaration

```
int my_number;
```

- Q: So, is `my_number` a number?

- A: Not really.
  - `my_number` is not a number, it's a variable
  - A number is fixed. `my_number` is a container to store a number
  - `my_number` is not (an arbitrary) number because it can't store numbers like 3.2 or π. It can only store integers

UCLA

Variables – Assignment

```
my_number = 5;
```

- This writes the value 5 into `my_number`.
- The = symbol is somewhat misleading. Something like

```
my_number <- 5;
```

  would be more clear (but this doesn't work)

- Again, strictly speaking, `my_number` is not 5, but "The (current) value of `my_number` is 5"
- The first assignment is also called *initialization*

Q: What value is stored in "`my_number`" before the first assignment?

A: No one knows. Accessing an uninitialized variable results in an "**undefined behavior**" and *anything can happen*. It's considered bad! Programmers don't want a situation that they can't control.

UCLA

Variables – Assignment

- Note that, in math, it would be wrong to write

$$x = 5$$
$$x = 10$$

- We can't assert x is 5 and then assert it is 10.

- On the other hand, in programming, there's nothing wrong about

```
my_number = 5;
my_number = 10;
```

- Again, the = symbol in C++ does not mean *equality*

UCLA

Useful digression: programming tip

I don't know what's going on. This is uncomfortable!

- Move on and first try to get things to work.
  (Sometimes, e.g. when studying math, this is a terrible attitude.)
- Just because there is a part you do not understand, doesn't mean you should stop to figure it out.
- First make things work, and then worry about other things.
- In serious programming, you'll never understand everything.

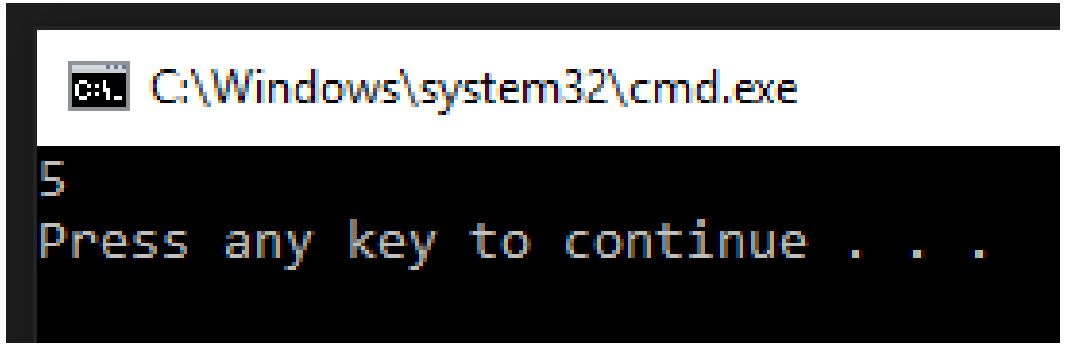
Slide from PIC10A (2017 Fall) taught by Prof. Ernest Ryu

The type `int` – operators

- Operators such as +, -, *, / are defined for the type `int`

```
int num1 = 4; // declartion & initialization at the same time
int num2 = 1;
int num3;
num3 = num1 + num2;
cout << num3 << endl;
```

- What is the output?



The type `int` – operators

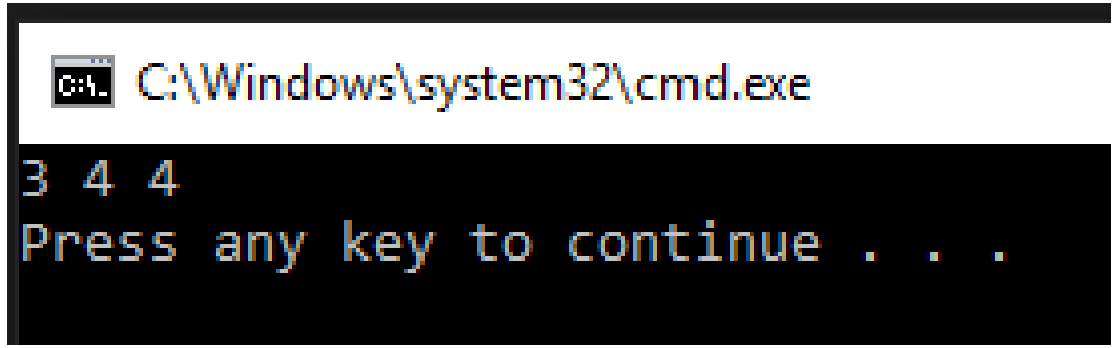
- Operators such as +, -, *, / are defined for the type `int`

```
int num4 = num1 - num2;
int num5 = num1 * num2;
int num6 = num1 / num2;
cout << num4 << ' ' << num5 << ' ' << num6 << endl;
```

Recall that:

```
int num1 = 4;
int num2 = 1;
```

- What is the output?



UCLA

The type `int` – operators

- Operators such as +, -, *, / are defined for the type `int`

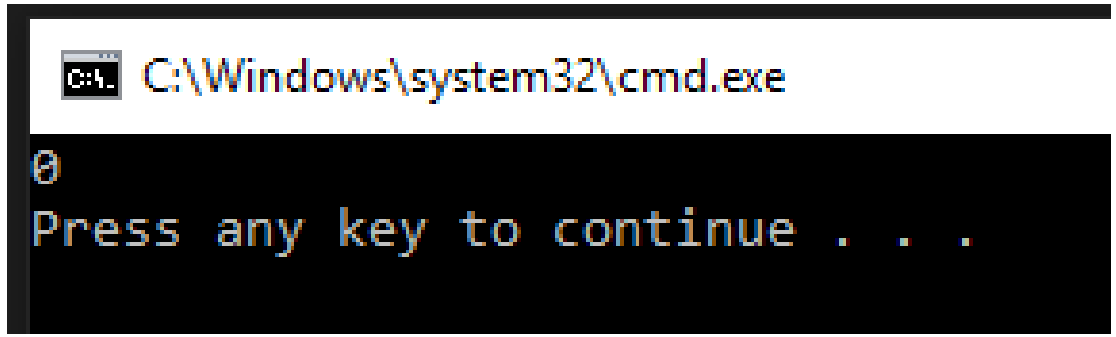
```
int num7 = num2 / num1;
cout << num7 << endl;
```

Recall that:

```
int num1 = 4;
int num2 = 1;
```

- What is the output?

Why isn't it 0.25?
More on this later…



UCLA

The type `int` – operators

- Q: So, operator "/" isn't exactly division?

- A: That's correct
  - num7 can't store 1/4 = 0.25, because it is of type `int`
  - Operator / for two `int`s performs division and take the integral part

```
int num1 = 11;
int num2 = 4;
cout << num1 / num2 << endl;
```

- What is the output?
- Use operator % for the remainder:

```
cout << num1 % num2 << endl;
```

UCLA

The type `int` – limitations

- Computers are finite. The type `int` has a maximum (and minimum) value

```
int num = 100;
cout << num << endl;
num = num * 100;
cout << num << endl;
num = num * 100;
cout << num << endl;
num = num * 100;
cout << num << endl;
num = num * 100;
cout << num << endl;
num = num * 100;
cout << num << endl;
```

- What is the output?

C:\Windows\system32\cmd.exe

```
100
10000
1000000
100000000
1410065408
-727379968
Press any key to continue . . .
```

UCLA

When poll is active, respond at **pollev.com/bumsukim297**

Text **BUMSUKIM297** to **37607** once to join

Can the following code overflow?

Yes, it may overflow when x is some special value.

No, it is safe.

Powered by Poll Everywhere

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

```
#include <iostream>
using namespace std;
int main() {
    int x = INT_MIN;
    cout << x << endl;
    cout << -x << endl;
    return 0;
}
```

Answer: Yes.
-INT_MIN is greater than INT_MAX!