PIC 10A 2B

TA: Bumsu Kim

UCLA

Today…

- String Manipulations
- HW 1 Hints
- Control Flows

Exercise (String Manipulation I)

- Q) What is the output for the following code?

```
string s1, s2, s3;
s1 = "PIC";
s2 = "10A";
s3 = s1 + s2;
s3 += "\n";
cout << s2.length() << " " << s3.length();
```

UCLA

Exercise (String Manipulation I)

- Q) What is the output for the following code?

```
string s1, s2, s3;
s1 = "PIC";
s2 = "10A";
s3 = s1 + s2;
s3 += "\n";
cout << s2.length() << " " << s3.length();
```

- A) 3 7

```
string s1, s2, s3;
s1 = "PIC";    // length == 3
s2 = "10A";
s3 = s1 + s2; // s3 == "PIC10A"    (length == 6)
s3 += "\n";    // s3 == "PIC10A\n" (length == 7)
cout << s2.length() << " " << s3.length();
```

UCLA

Exercise (String Manipulation II)

- Q) Consider the following program.

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

int main() {
  string s = "a more perfect union";
  cout << s[1];
  cout << s[3];
  cout << s[5] << "\n";
}
```

What is the output?

A. oe
B. a m
C. amo
D. Neither of the above

Exercise (String Manipulation II)

- Q) Consider the following program.

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

int main() {
  string s = "a more perfect union";
  cout << s[1];
  cout << s[3];
  cout << s[5] << "\n";
}
```

What is the output?

A. oe
B. a m
C. amo
D. Neither of the above

A) A.
C++ indexing always starts from 0

UCLA

Exercise – Number to Month Names

- This exercise is from the textbook

Exercise P2.19. Write a program that transforms numbers `1`, `2`, `3`, …, `12` into the corresponding month names `January`, `February`, `March`, …, `December`. *Hint:* Make a very long string `"January February March ..."`, in which you add spaces such that each month name has *the same length*. Then use `substr` to extract the month you want.

UCLA

VS Tips: How to run the code you downloaded

- Need to know when you
  - download code from my Github repo
  - or you want to test your own code in PIC Lab
- 1. Download the file
- 2. Create an empty project (with a descriptive name)
- 3. Add the file to the folder “Source Files” on the “Solution Explorer” window
- 4. If there are several projects in your solution (recommended!), set the new project “as Startup Project”
- 5. Run!

UCLA

VS Tips: How to run the code you downloaded

- Need to know when you
  - download code from my Github repo
  - or you want to test your own code in PIC Lab
- 1. Download the file
- 2. Create an empty project (with a descriptive name)
- 3. Add the file to the folder “Source Files” on the “Solution Explorer” window
- 4. If there are several projects in your solution (recommended!), set the new project “as Startup Project”
- 5. Run!

UCLA

Homework 1 Hints

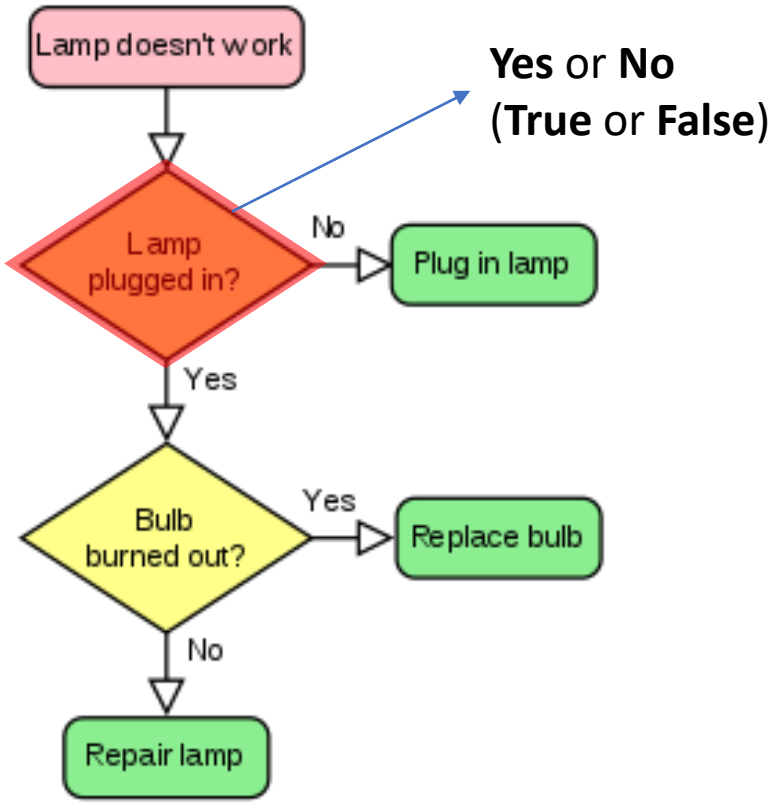
- Problem 1: Not much to say. Just be careful and double check the output!
- Problem 2 is a little more difficult
- Strategy:
  - Extract each digit first (See the exercise from last Tuesday)
  - Construct a “reversed” number
- Don’t get the user input as a string and extract digits as characters!
  - In most cases it should be fine
  - But in some cases, it will output wrong answers (consider “100”)

- Finally, read the instructions carefully, and follow it
  - File names, ownership declaration, coding practice and comments, etc.
  - *Watch the “.” at the end!!!!*

UCLA

Control Flow – `if` / `else` / `else if`

- Flowchart?



- Sometimes your program should run differently according to the current state
- In the world of programming, every question is basically a yes-no question
  - It's quite natural if you recall how the computer works; it's always binary. A light bulb (*bit*) can only be *on(1)/off(0)*
- More formally, the program evaluates an expression and determines if it is `true` or `false`
- Recall the "bool" (Boolean) type:

```
bool true_or_false; // bool literals: true and false
```

UCLA

The Type bool

- The type bool is one of the *fundamental types* in C++

- bool uses 1 byte (not 1 bit!) to store a logical value, "true" or "false".
  - true and false are bool literals

```
Microsoft Visual Studio Debug Console
1
true
```

- Comparison operators return bool values

```
5 < 3;
(bool)false
```

```
int a = 5, b = 3;
bool comparison = (a >= b);
cout << comparison << endl;
```

```
Microsoft Visual Studio Debug Console
1
```

```
int a = 5, b = 3;
bool comparison = (a >= b);
cout << comparison << endl;

cout << boolalpha;
cout << comparison << endl;
```

UCLA

The Type `bool`

- Every numeric value other than 0 is implicitly casted to `true`
  - 0 is false
- Other expressions may or may not be convertible to `bool`
  - e.g. pointers. More on this later…

- Just like operators +, –, /, * defined for the numeric types, there are operators designed for Boolean variables, called **logical operators**
  - Some of the common logical operators are NOT(unary), AND(binary), and OR (binary)
  - e.g.
    - **(NOT(** `true` **))** evaluates `false`
    - **(** `true` **AND** `false` **)** evaluates `false`, and **(** `true` **AND** `true` **)** evaluates `true`
    - **(** `false` **OR** `true` **)** evaluates `true`, etc.
  - Operator NOT : ! → ! (`true`) stands for **NOT(** `true` **)**
  - Operator AND : && → (`true && false`) stands for **(** `true` **AND** `false` **)**
  - Operator OR : || → (`false || true`) stands for **(** `false` **OR** `true` **)**

UCLA

Boolean Algebra

- Recall that
  - `A && B` is true only when A and B are both true
  - `A || B` is false only when A and B are both false
- De Morgan's law:
  - `!(A && B) == (!A) || (!B)`
  - `!(A || B) == (!A) && (!B)`

Truth Tables

&&		P	
		T	F
Q	T	T	F
	F	F	F

||		P	
		T	F
Q	T	T	T
	F	T	F

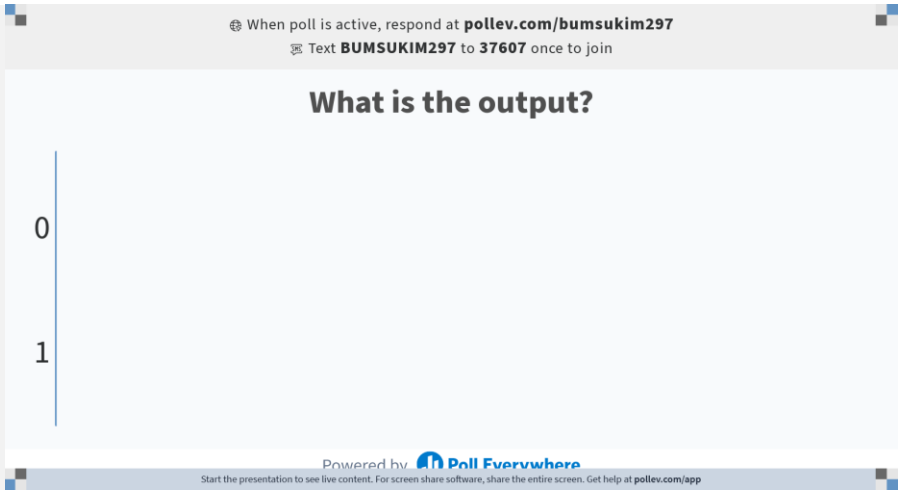
UCLA

Exercise (Boolean Algebra)

2. Consider the following program.

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

int main() {
  bool b1 = true, b2 = false;
  cout << ( !(b1 && !b2) || b2 ) << endl;
}
```



What is the output?

A. 0

B. 1

- Recall that `true` outputs 1, and `false` outputs 0 on the console
  - (Can be strings "true" and "false," respectively, if you use the option `std::boolalpha`)

Exercise (Boolean Algebra)

```
bool b1 = true, b2 = false;
```

```
!(b1 && !b2) || b2
```

True

True

False

&& (AND)

True

! (not)

OR

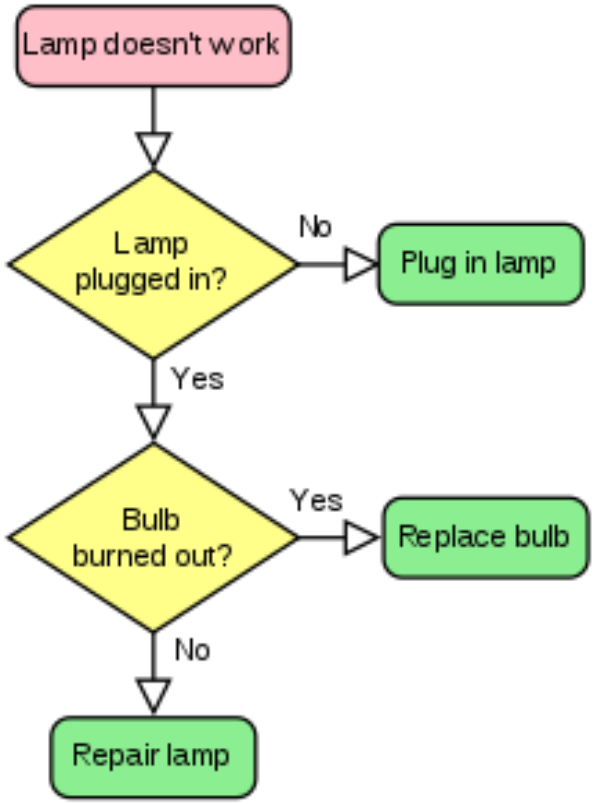
False

False

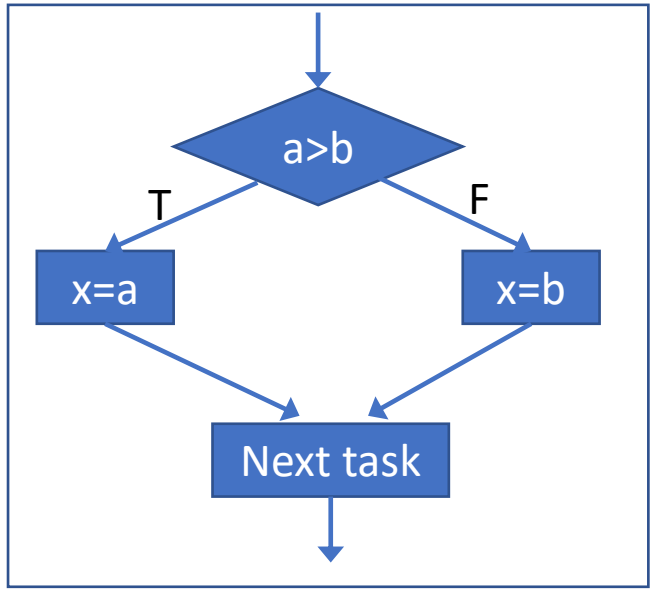
UCLA

Control Flow – `if` / `else` / `else if`

- Now you're ready to implement a flow chart like
- Only need to use "`if`" and "`else`"



- For instance, if you want `x = max(a, b)`, the flow char would look like the following:



- The implementation is simple:

```
if (a > b) { x = a; }
else { x = b; }
```

UCLA

A Useful Digression – Coding Style

- Recall that C++ is quite generous about the white spaces
- It is important, however, to follow a ***standard*** coding style for readability
- So, while

```
if (a > b) { x = a; }
else { x = b; }
```

or even

```
if (a > b) x = a; else x = b;
```

Without curly braces!

is (grammatically) allowed, it is better to write it as either

```
if (a > b) {
      x = a;
} else {
      x = b;
}
```

or

```
if (a > b)
{
      x = a;
}
else
{
      x = b;
}
```

UCLA

A Useful Digression – Coding Style

- For instance, Google has its own C++ style guide:

Each of `if`, `else`, and `else if` belong on separate lines. There should be a space between the `if` and the open parenthesis, and between the close parenthesis and the curly brace (if any), but no space between the parentheses and the condition.

```
if (condition) {  // no spaces inside parentheses
  ...  // 2 space indent.
} else if (...) {  // The else goes on the same line as the closing brace.
  ...
} else {
  ...
}
```

Google C++ Style Guide: https://google.github.io/styleguide/cppguide.html#Conditionals

```
if(condition) {   // Bad - space missing after IF.
if ( condition ) { // Bad - space between the parentheses and the condition
if (condition){   // Bad - space missing before {.
if(condition){    // Doubly bad.
```

- Conclusion: Good coding style is important!

UCLA

Control Flow – `if` / `else` / `else if`

- Good coding practice: always use curly braces {} with control statements
- Even when you only need to execute a single expression

Bad

```
if (a > b)
    x = a;
else
    x = b;
```

vs

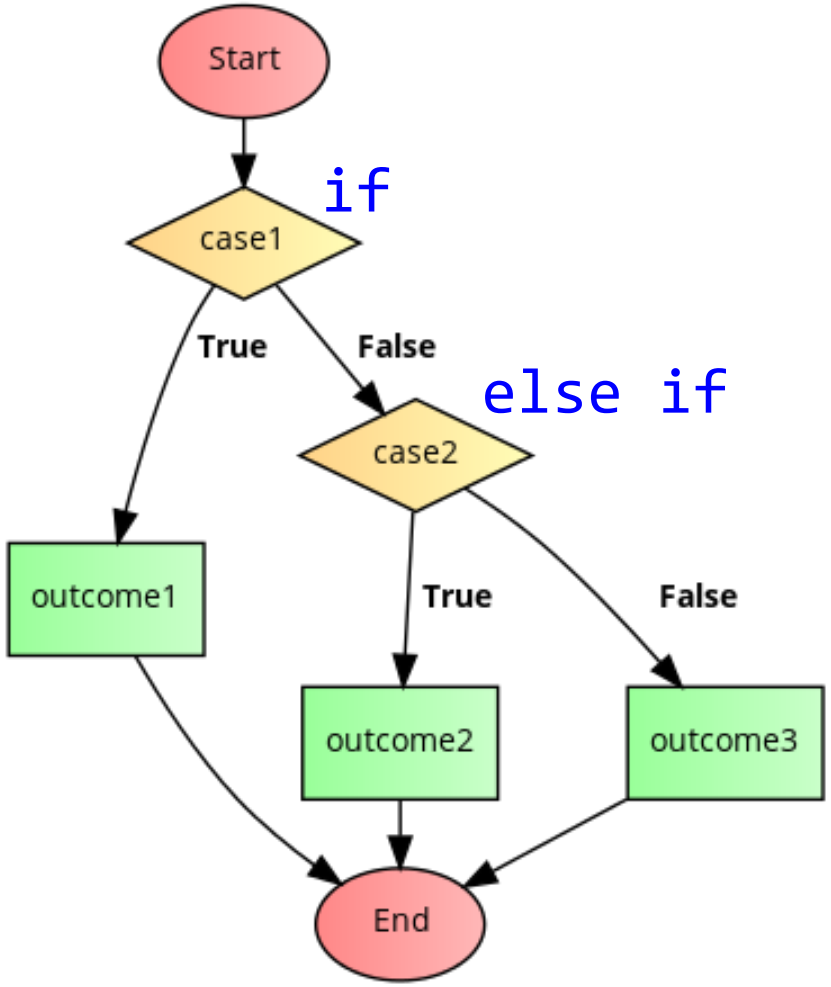
```
if (a > b) {
    x = a;
} else {
    x = b;
}
```

Good

UCLA

Control Flow – `if` / `else` / `else if`

- `else if` is just a compound form of `else{ if ( … ) }`



Implementation

```
if (case1) {
      outcome1;
}
else if (case2) {
      outcome2;
}
else {
      outcome3;
}
```

Exercise (Control Flow – If/Else)

7. Consider the following program, which is poorly indented on purpose.

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

int main() {
if (true)
cout << "a";
else
cout << "b";
cout << "c";
}
```

What is the output?

A. a B. b C. c D. ab E. ac F. bc G. abc

UCLA

Exercise (Control Flow – If/Else)

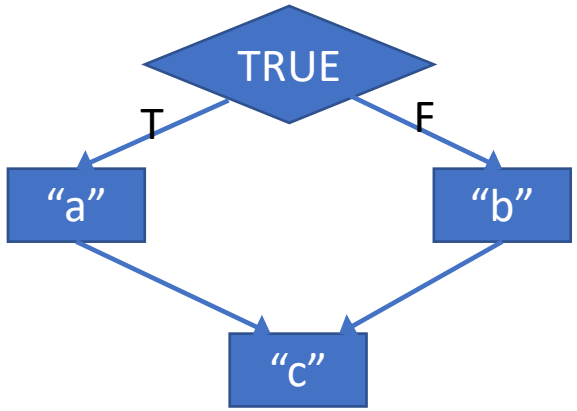
7. Consider the following program, which is poorly indented on purpose.

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

int main() {
if (true)
cout << "a";
else
cout << "b";
cout << "c";
}
```

What is the output?

A. a B. b C. c D. ab E. ac F. bc G. abc



UCLA

Exercise (Quadratic Formula)

Exercise P3.1. Write a program that prints all solutions to the quadratic equation $ax^2 + bx + c = 0$. Read in a, b, c and use the quadratic formula. If the discriminant $b^2 - 4ac$ is negative, display a message stating that there are no solutions.

Hint 1: The quadratic formula is

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Hint 2: Use a "if – else if – else" structure.
The number of solutions can be 0, 1, or 2, depending on the discriminant.

UCLA

Your Feedback is welcome

- Don’t hesitate to give a feedback on the discussion
- Use the link on my Github repo, or the link below:
  - https://forms.gle/erZj1iSgHNrHQuXk6

My Github repo on the web looks like:

code	Week2 Tu
LICENSE	Initial commit
README.md	Update README.md

README.md

PIC10A

PIC10A discussion 2B, UCLA for Fall 2022

Google form link for feedbacks: https://forms.gle/erZj1iSgHNrHQuXk6

Click this link

UCLA