CS 31 Week 2 Discussion 2E

Srinath

Announcements

- Project 2 warmup is up! Due **11:00 PM, Saturday, October 8**
- Project 2 is also up! Due **11:00 PM, Thursday, October 13**
- For project 2, make sure to check the accompanying homework :)

Outline

- Learnings so far
- Strings
- If statements
- Operators
- Worksheet 2

Learnings so far…

Variable types

Variable declarations with and without initialization

Arithmetic expressions

If statements

Assignment statements

Learnings so far…

Variable types

```
int, double, string
```

Variable declarations with and without initialization

```
int a;
int a = 100;
```

Arithmetic expressions

```
+, -, *, /
2+9, a-b
```

If statements

```
if(condition) {do something..}
```

Assignment statements

```
a = 190;
a = b;
```

Strings

Strings :

Strings consists of a sequence of characters.
Eg : “hello world”
Empty string : “”

Strings :

Strings consists of a sequence of characters.
Eg : “hello world”
Empty string : “”
“ ” – is this empty string?

Strings :

Strings consists of a sequence of characters.
Eg : “hello world”
Empty string : “”
“ ” – is this empty string? - No, it has SPACE characters.

Strings :

Strings consists of a sequence of characters.
Eg : “hello world”
Empty string : “”
“ ” – is this empty string? - No, it has SPACE characters.

Does your program by default know what a string data type is?

Strings :

Strings consists of a sequence of characters.
Eg : “hello world”
Empty string : “”
“ ” – is this empty string? - No, it has SPACE characters.

Does your program by default know what a string data type is?

- No, string is not primitive type like int. Is part of some library like std.
- using namespace std;, std::string

Strings :

Strings consists of a sequence of characters.
Eg : “hello world”
Empty string : “”
“ ” – is this empty string? - No, it has SPACE characters.

Does your program by default know what a string data type is?

- No, string is not primitive type like int. Is part of some library like std.
- using namespace std;, std::string

What kind of error will you get if you don’t include any string library and yet use string type?

Strings :

Strings consists of a sequence of characters.
Eg : “hello world”
Empty string : “”
“ ” – is this empty string? - No, it has SPACE characters.

Does your program by default know what a string data type is?

- No, string is not primitive type like int. Is part of some library like std.
- using namespace std;, std::string

What kind of error will you get if you don’t include any string library and yet use string type?

- A compilation error, program doesn’t know what a ‘string’ data type is.

Strings : Allowed operations

```
string s = “hello”;  // create and initialize one string.

string t = s; // create another string with the same value as s.

string u; // create a string and initialize it to empty string.

u = s; // assign strings.

if (s==”hello”) // compare strings with ==.
```

Strings : Reading in

How to read in a string from user input?

Strings : Reading in

How to read in a string from user input?

- Use **getline(cin, s);**
- Let's say the user typed "Hello world" and then Enter(i,e newline)

Strings : Reading in

How to read in a string from user input?

- Use **getline(cin, s);**
- Let's say the user typed "Hello world" and then Enter(i,e newline)
- All of the string along with newline will be consumed by the program i,e "Hello world\n"
- But, **s** doesn't contain newline. i,e **s** will be "Hello world"

Strings : Reading in

How to read in a string from user input?

- Use **getline(cin, s);**
- Let's say the user typed "Hello world" and then Enter(i,e newline)
- All of the string along with newline will be consumed by the program i,e "Hello world\n"
- But, **s** doesn't contain newline. i,e **s** will be "Hello world"

cin.ignore(10, '\n');

What does this do?

Strings : Reading in

How to read in a string from user input?

- Use **getline(cin, s);**
- Let's say the user typed "Hello world" and then Enter(i,e newline)
- All of the string along with newline will be consumed by the program i,e "Hello world\n"
- But, **s** doesn't contain newline. i,e **s** will be "Hello world"

cin.ignore(10, '\n');

What does this do?

- Ignores characters available to program until '\n' is reached or a maximum of 10 characters.

Strings : Reading in

How to read in a string from user input?

- Use **getline(cin, s);**
- Let's say the user typed "Hello world" and then Enter(i,e newline)
- All of the string along with newline will be consumed by the program i,e "Hello world\n"
- But, **s** doesn't contain newline. i,e **s** will be "Hello world"

cin.ignore(10, '\n');

What does this do?

- Ignores characters available to program until '\n' is reached or a maximum of 10 characters.
- 'abcd\n' - all of it will be ignored
- 'abcdefghijkl\n' - only 'abcdefghij' will be ignored.

Strings : Reading in

cin.ignore(10, ‘\n’);

What does this do?

- Ignores characters available to program until ‘\n’(including it) is reached or a maximum of 10 characters.
- ‘abcd\n’ - all of it will be ignored
- ‘abcdefghijkl\n’ - only ‘abcdefghij’ will be ignored.

When should we use it?

Strings : Reading in

cin.ignore(10, '\n');

What does this do?

- Ignores characters available to program until '\n'(including it) is reached or a maximum of 10 characters.
- 'abcd\n' - all of it will be ignored
- 'abcdefghijkl\n' - only 'abcdefghij' will be ignored.

When should we use it?

- *only* time to use is after reading a number and the next to read is a string.

```
cout << "How many courses you enrolled this term?"<<endl;
int numCoursesEnrolled;
cin >> numCoursesEnrolled;
cin.ignore(10000, '\n');

cout << "What is your favourite course among them?"<<endl;
string favCourse;
getline(cin, favCourse);
```

Strings : Reading in

cin.ignore(10, '\n');

What does this do?

- Ignores characters available to program until '\n'(including it) is reached or a maximum of 10 characters.
- 'abcd\n' - all of it will be ignored
- 'abcdefghijkl\n' - only 'abcdefghij' will be ignored.

When should we use it?

- *only* time to use is after reading a number and the next to read is a string.

```
cout << "How many courses you enrolled this term?"<<endl;
int numCoursesEnrolled;
cin >> numCoursesEnrolled;
cin.ignore(10000, '\n');

cout << "What is your favourite course among them?"<<endl;
string favCourse;
getline(cin, favCourse);
```

What will happen if we don't use it?

Strings : Reading in

cin.ignore(10, '\n');

What does this do?

- Ignores characters available to program until '\n'(including it) is reached or a maximum of 10 characters.
- 'abcd\n' - all of it will be ignored
- 'abcdefghijkl\n' - only 'abcdefghij' will be ignored.

When should we use it?

- *only* time to use is after reading a number and the next to read is a string.

```
cout << "How many courses you enrolled this term?"<<endl;
int numCoursesEnrolled;
cin >> numCoursesEnrolled;
cin.ignore(10000, '\n');

cout << "What is your favourite course among them?"<<endl;
string favCourse;
getline(cin, favCourse);
```

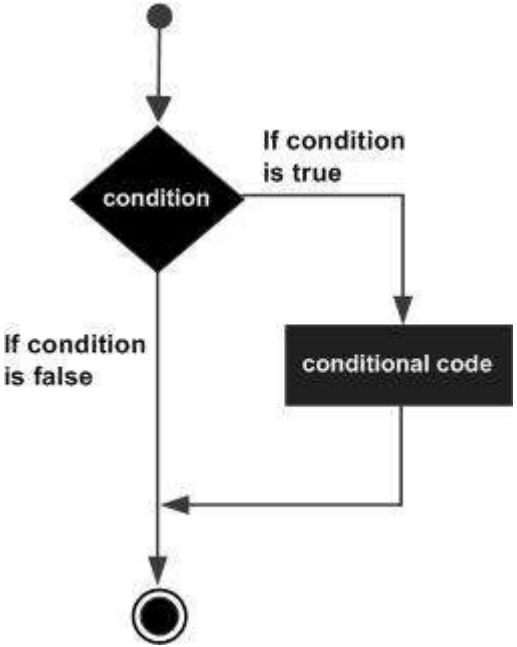
What will happen if we don't use it?

- Empty string will be read into the variable favCourse because '\n' entered for reading the int is still available to the program.

If statements

If statements : If …

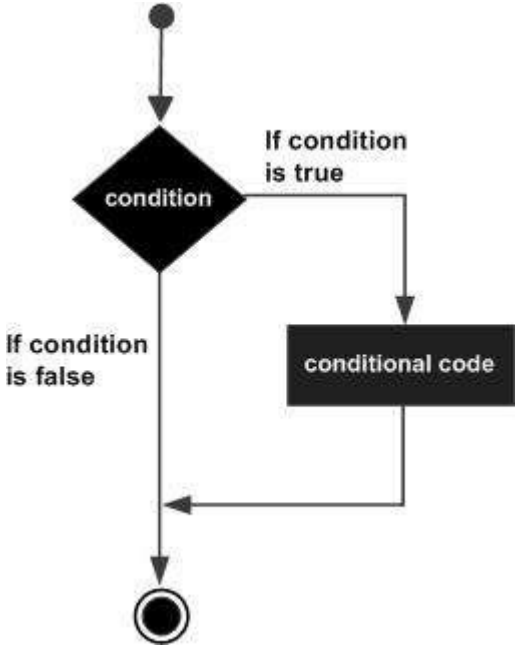
```
if(boolean_expression) {
   // statement(s) will execute if the boolean expression is true
}
```



If statements : If ...

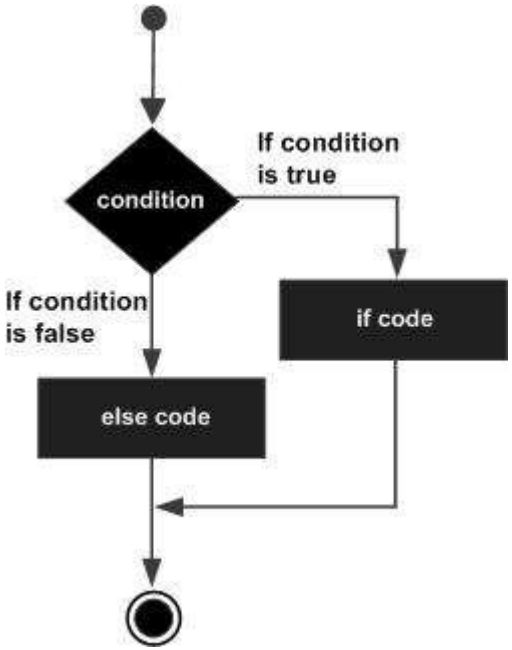
```
if(boolean_expression) {
   // statement(s) will execute if the boolean expression is true
}
```

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

int main () {
   // local variable declaration:
   int a = 10;

   // check the boolean condition
   if( a < 20 ) {
      // if condition is true then print the following
      cout << "a is less than 20;" << endl;
   }
   cout << "value of a is : " << a << endl;

   return 0;
}
```



If statements : If … else …

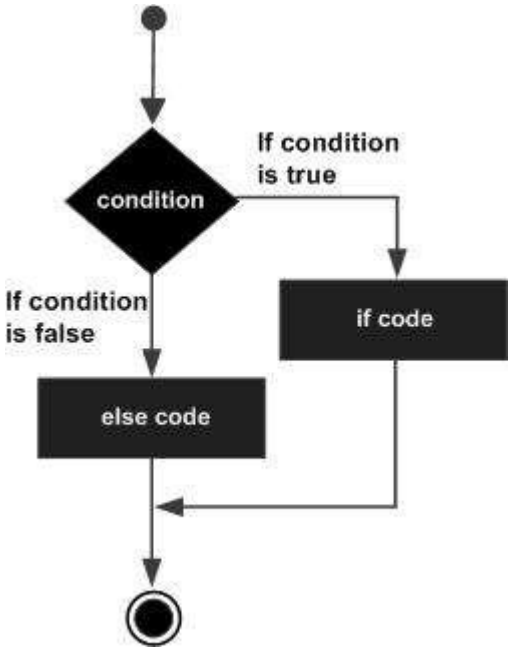
```
if(boolean_expression) {
   // statement(s) will execute if the boolean expression is true
} else {
  // statement(s) will execute if the boolean expression is false
}
```



If statements : If … else …

```
if(boolean_expression) {
   // statement(s) will execute if the boolean expression is true
} else {
  // statement(s) will execute if the boolean expression is false
}
```

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

int main () {
   // local variable declaration:
   int a = 100;

   // check the boolean condition
   if( a < 20 ) {
      // if condition is true then print the following
      cout << "a is less than 20;" << endl;
   } else {
      // if condition is false then print the following
      cout << "a is not less than 20;" << endl;
   }
   cout << "value of a is : " << a << endl;

   return 0;
}
```



If statements : If … else if …. else

```
if(boolean_expression 1) {
   // Executes when the boolean expression 1 is true
} else if( boolean_expression 2) {
   // Executes when the boolean expression 2 is true
} else if( boolean_expression 3) {
   // Executes when the boolean expression 3 is true
} else {
   // executes when the none of the above condition is true.
}
```

If statements : If … else if …. else

```
if(boolean_expression 1) {
   // Executes when the boolean expression 1 is true
} else if( boolean_expression 2) {
   // Executes when the boolean expression 2 is true
} else if( boolean_expression 3) {
   // Executes when the boolean expression 3 is true
} else {
   // executes when the none of the above condition is true.
}
```

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

int main () {
   // local variable declaration:
   int a = 100;

   // check the boolean condition
   if( a == 10 ) {
      // if condition is true then print the following
      cout << "Value of a is 10" << endl;
   } else if( a == 20 ) {
      // if else if condition is true
      cout << "Value of a is 20" << endl;
   } else if( a == 30 ) {
      // if else if condition is true
      cout << "Value of a is 30" << endl;
   } else {
      // if none of the conditions is true
      cout << "Value of a is not matching" << endl;
   }
   cout << "Exact value of a is : " << a << endl;

   return 0;
}
```

Operators

Operators : Comparison operators

operator	description
==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

```
(7 == 5)      // evaluates to false
(5 > 4)       // evaluates to true
(3 != 2)      // evaluates to true
(6 >= 6)      // evaluates to true
(5 < 5)       // evaluates to false
```

Operators : Comparison operators

operator	description
==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

```
(7 == 5)      // evaluates to false
(5 > 4)       // evaluates to true
(3 != 2)      // evaluates to true
(6 >= 6)      // evaluates to true
(5 < 5)       // evaluates to false
```

Suppose a=2, b=3, c=6

```
(a == 5)      // evaluates to false, since a is not equal to 5
(a*b >= c)    // evaluates to true, since (2*3 >= 6) is true
(b+4 > a*c)   // evaluates to false, since (3+4 > 2*6) is false
((b=2) == a)  // evaluates to true
```

Operators : Logical operators

'!' - NOT operator, negation

```
!(5 == 5)    // evaluates to false because the expression at its right (5 == 5) is true
!(6 <= 4)    // evaluates to true because (6 <= 4) would be false
!true        // evaluates to false
!false       // evaluates to true
```

'&&' - AND operator

&& OPERATOR (and)		
a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

'||' - OR operator

|| OPERATOR (or)		
a	b	a || b
true	true	true
true	false	true
false	true	true
false	false	false

```
( (5 == 5) && (3 > 6) )  // evaluates to false ( true && false )
( (5 == 5) || (3 > 6) )  // evaluates to true ( true || false )
```

Operators : Short-Circuit

Only evaluate what is necessary from left to right.

Eg : ((5==5) || (3>6))

C++ first evaluates whether 5==5 is true. If so, it doesn’t check 3>6.

Operators : Short-Circuit

Only evaluate what is necessary from left to right.

Eg : ((5==5) || (3>6))

C++ first evaluates whether 5==5 is true. If so, it doesn't check 3>6.

operator	short-circuit
`&&`	if the left-hand side expression is `false`, the combined result is `false` (the right-hand side expression is never evaluated).
`||`	if the left-hand side expression is `true`, the combined result is `true` (the right-hand side expression is never evaluated).

Questions??

Output precision

Project 2 might require you to output values exactly to 2 decimal places.

This how we can achieve it in C++.

```
cout.setf(ios::fixed);
cout.precision(2);
```

Week 2 LA Slides

Liah Kim

LA Worksheet Overview

- Optional but highly recommended!
- Available on course website under “LA Worksheets”
- Solutions posted later on course website
- Two sections: reading and programming problems
- For now on Zoom
  - Collaborate and work through problems in breakout rooms
  - Each group will be responsible for presenting out their solution to one of the problems (super short/low stakes!)
  - Come back into the main room and go over problems together
- **Reading problems 1 and 3**
- **Programming problems 1 and 2**
- If you finish early, try the other problems!

Breakout Rooms

Please start by introducing yourself to each other!

- Name
- Year
- Major
- Favorite dining hall

Reading Problems

Problem 1

Assume that the following lines of code are inside the main function, with `#include <iostream>` and `using namespace std`, and all the string variables used have been previously declared.

Problem 1a

Highlight where you think the bug occurs

```
string name;
cout << "Enter your name: ";
getline(cin, name);

int UID;
string major;

cout << "\nEnter your UID: ";
cin >> UID ;

cout << "\nEnter your Major: ";
getline(cin, major);

cout << "\nEnter your residence hall: "
getline(cin, hall);

cout << "\n" << UID << " is the ID of " << name
     <<", a " << major << " student who lives in " << hall << endl;
```

Problem 1a Solution

Bug: It will skip “Enter your Major”, because getline has already consumed a newline character.

```
string name;
cout << "Enter your name: ";
getline(cin, name);

int UID;
string major;

cout << "\nEnter your UID: ";
cin >> UID;

cout << "\nEnter your Major: ";
getline(cin, major);

cout << "\nEnter your residence hall: ";
getline(cin, hall);

cout << "\n" << UID << " is the ID of " << name
     <<", a " << major << " student who lives in " << hall << endl;
```

Problem 1b

Explain what you think will happen when this program runs.

```
string name;
cout << "Enter your name: ";
getline(cin, name);

int UID;
string major;

cout << "\nEnter your UID: ";
cin >> UID ;

cout << "\nEnter your Major: ";
getline(cin, major);

cout << "\nEnter your residence hall: "
getline(cin, hall);

cout << "\n" << UID << " is the ID of " << name
     <<", a " << major << " student who lives in " << hall << endl;
```

Problem 1b Solution

- A newline is always appended to your input when you select Enter or Return when submitting from a terminal
  - cin discards any leading whitespace (blanks, tabs, newlines) and does *not* consume the character after the last digit
- When the flow of control reaches std::getline(), the newline will be discarded, but the input will cease immediately
  - The reason this happens is because the default functionality of this function dictates that it should (it attempts to read a line and stops when it finds a newline)

Problem 1c

Is this a logic or a compilation error? Why

```
string name;
cout << "Enter your name: ";
getline(cin, name);

int UID;
string major;

cout << "\nEnter your UID: ";
cin >> UID ;

cout << "\nEnter your Major: ";
getline(cin, major);

cout << "\nEnter your residence hall: "
getline(cin, hall);

cout << "\n" << UID << " is the ID of " << name
     <<", a " << major << " student who lives in " << hall << endl;
```

Problem 1c Solution

This is more of a logic error than a compilation error because it will not stop your program from compiling. However, because we have made a mistake in predicting what the computer will do, we have made an error in our logic. Errors like this can sneak up on you without the compiler letting you know.

Problem 1d

Add a fix to the problem you found in part (a).

```
string name;
cout << "Enter your name: ";
getline(cin, name);

int UID;
string major;

cout << "\nEnter your UID: ";
cin >> UID;

cout << "\nEnter your Major: ";
getline(cin, major);

cout << "\nEnter your residence hall: ";
getline(cin, hall);

cout << "\n" << UID << " is the ID of " << name
     <<", a " << major << " student who lives in " << hall << endl;
```

Problem 1d Solution

- Because this leading newline inhibits the expected functionality of your program, it follows that it must be skipped or ignored somehow.
  - One option is to call `cin.ignore(10000, '\n')` after the first extraction. It will discard characters up to the next newline so that the newline is no longer intrusive.

* key takeaway: if you use cin to get an integer input and the next input will be a string, make sure to follow it with cin.ignore()

Problem 1d Solution Code

```
string name;
cout << "Enter your name: ";
getline(cin, name);

int UID;
string major;

cout << "\nEnter your UID: ";
cin >> UID;
cin.ignore(10000, '\n');          // add this

cout << "\nEnter your Major: ";
getline(cin, major);

cout << "\nEnter your residence hall: ";
getline(cin, hall);

cout << "\n" << UID << " is the ID of " << name
     <<", a " << major << " student who lives in " << hall << endl;
```

Problem 3

This code snippet takes a certain “hour” and “weekday” and tries to tell you if you can buy turnips from Daisy Mae, the turnip seller.

```
int hour;
string weekday;
cin >> hour;
cin >> weekday;

if (weekday != "Sunday" || hour >= 12) {
   cout >> "Daisy Mae is not here!" >> endl;
} else {
   if (hour = 11) {
       cout >> "It's almost 12! Hurry up!" >> endl;
   } else {
       cout >> "Buy turnips with Bells." >> endl;
}
```

Problem 3a

Find the 5 lines with mistakes in the code and fix them.

```
1 int hour;
2 string weekday;
3 cin >> hour;
4 cin >> weekday;
5
6 if (weekday != "Sunday" || hour >= 12) {
7    cout >> "Daisy Mae is not here!" >> endl;
8 } else {
9    if (hour = 11) {
10       cout >> "It's almost 12! Hurry up!" >> endl;
11   } else {
12       cout >> "Buy turnips with Bells." >> endl;
13 }
```

Problem 3a Solution

The arrows for the output statements are in the wrong direction (should be << instead of >>). There should also be a double equal (==) in the if statement. There is also a missing closing bracket (}) on the else statement.

```
if (weekday != "Sunday" || hour >= 12) {
   cout >> "Daisy Mae is not here!" >> endl;
} else {
   if (hour = 11) {
       cout >> "It's almost 12! Hurry up!" >> endl;
   } else {
       cout >> "Buy turnips with Bells." >> endl;
    } (was not in the program originally!)
}
```

Problem 3b and 3c

(b) Will this code compile? Why or why not?

(c) After you fix the bugs, imagine you input 11 for the hour, then “Monday” for the weekday. What will this program say to you?

Problem 3b and 3c Solution

- 3b
  - This code will not compile!
  - The line with the conditional (hour = 11) will not be a compile error but would be a logic error because there will be a variable assignment instead of an expected truth value.
  - All of the other errors will cause compilation errors
- 3c
  - Daisy Mae is not here!

Programming Problems

Problem 1

Write a program that asks for a number between 0 and 100 (exclusive), and takes an integer input. If you input a number greater than or equal to 100, it will print “Liar, liar, plants for hire” and stop. If you input a number less than or equal to 0, it will print “Liar, liar, plants for hire” and stop. If your number has a “tens” digit that would round up to 100, meaning the number is at least 50, and will print “Almost to 100!” Otherwise, it will print “Still a-ways to go!”

(Example output of this program on next slide)

Problem 1

Example Output:

<u>Case one:</u>
Please give me a number between 0 and 100
0
Liar, liar, plants for hire.

<u>Case two:</u>
Please give me a number between 0 and 100
95
Almost to 100!

<u>Case three:</u>
Please give me a number between 0 and 100
45
Still a-ways to go!

Problem 1 Solution

```
int num;

// Print the statement and take the input
cout << "Please give me a number between 0 and 100." << endl;
cin >> num;

// We can check each end in case the number is out of range.
if (num > 99 || num < 1) {
    cout << "Liar, liar, plants for hire." << endl;
} else {
    int tens = num / 10;

    if (tens >= 5) {
        cout << "Almost to 100!" << endl;
    } else {
        cout << "Still a-ways to go!" << endl;
    }
}
```

Problem 2

Write a program that takes in two numbers and a command of type string ("Add", "Subtract", "Multiply", "Divide"). Inputting an invalid command should cause the program to print out “Invalid command!” and stop.

Sample output:

Enter your first number: 3

Enter your second number: 7

Enter your command: Multiply

Result: 21

Problem 2 Solution

```
int first = 0;
int second = 0;
string command = "";

cout << "Enter your first number: ";
cin >> first;

cout << "Enter your second number: ";
cin >> second;
// ignore the next \n to avoid complications!
cin.ignore(1000000, '\n');

cout << "Enter your command: ";
getline(cin, command);

// Check the string to each possible type of command
if (command == "Add")
      cout << "Result: " << first + second << endl;
else if (command == "Subtract")
      cout << "Result: " << first - second << endl;
else if (command == "Multiply")
      cout << "Result: " << first * second << endl;
else if (command == "Divide" && second != 0)
      cout << "Result: " << first / second << endl;
else {
      cout << "Invalid command!" << endl;
}
```

LA Feedback Form

- Anonymous (if you want it to be!)
- If you have any questions, comments, concerns, you can voice them here
- https://forms.gle/xm85LnhXaz8bcVNa9
- You can always email me for specific concerns/requests :)
    - liahk1@g.ucla.edu

Questions?