#include <stdio.h>
int main(void)
{
    int count;
    for (count = 1; count <= 500; count++)
        printf("I will not throw paper airplanes in class.\n");
    return 0;
}
We put a loop around code that we want to run more than once.

If we have an easy sequence (0, 1, 2, ... 10) of values we want to go over, for loop is nice.

Otherwise, the while loop is a bit more general and is typically more useful if we are asking the user to control the loop.
Write a program that asks the user to input a value, then show the sum from 1 to that value in the following format:

Find the sum from 1 to what value?  5

1+2+3+4+5  =  15

(See: sumToN.cpp)
Nested for loop

Now modify the code so it shows all sums less than or equal to the entered values, as such:

Find the sum from 1 to what value? 4
1 = 1
1+2 = 3
1+2+3 = 6
1+2+3+4 = 10
(See: sumAllToN.cpp)
Nested for loop

Like nested if statements, we can also make nested loops (which can cause headaches)

It might help to think of each loop as an added dimensions:

1 loop = 1 dimension (line/ruler)
2 loops = 2 dimensions (plane/square/area)
3 loops = 3 dimensions (volume/cube)
...
(See: nestedLoop.cpp)
Nested for loop

Ask the user for a size of matrix, then show the identity matrix for that dimension:

What size? 4
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1

(See: identityMatrix.cpp)
do-while loop

A do-while loop is similar to a normal while loop, except the bool expression is only tested at the end of the loop (not at the start)

```cpp
cout << "How many times do you want to run the loop?\n";  
cin >> i; // what happens if i is less than 1?  
do {  
    cout << "Looping, i = " << i << "\n";  
    i--;  
} while (i > 0);  
cout << "Outside the loop, i = " << i << "\n";
```

(See: doWhile.cpp)
do-while loop

Q: Why would I ever want a do-while loop?

A: When the first time the variable is set is inside the loop.

You can initialize the variable correctly and use a normal while loop, but this makes the logic harder.
while vs do-while

(see: vendingV2.cpp)