Even and Odd numbers

This is a summary of yesterday's introduction of the mod function (%). In C, the function % gives us the remainder. For example:

$$\begin{aligned} 8\%5 = 3, &\qquad \text{when we divide 8 by 5, there are 3 remainder} \\ 10\%3 = 1, &\qquad \text{when we divide 10 by 3, there is 1 remainder} \end{aligned}$$

There is a special case of no remainder, as then one number fits exactly into another. If we divide a by b and have no remainder, we say that b *divides* a or that b is a *factor* of a.

For example, 3 divides 12 (or 3 is a factor of 12) because $12/3 = 4$ with no remainder, but 5 does not divide 12 because $12/5 = 2$ remainder 2. We can also write this as

$$12\%3 = 0, \qquad 12\%5 = 2$$

Yesterday we determined that a number was even if is was divisible by 2, otherwise it was odd. That is, if N is our number, there were two cases:

$$N\%2 = \begin{cases} 0 & N \text{ divisible by 2, so } N \text{ is even} \\ 1 & N \text{ not divisible by 2, so } N \text{ is odd} \end{cases}$$

We then were in a position to write a program (on the reverse side of this page) that checked if a number N was even. Our attack was

1. Ask the user for a number, store the result in N
2. Check if $N\%2$ is zero
   (a) If $N\%2 == 0$ (remember two equals signs makes a question in C!) then tell the user their number is even
   (b) If $N\%2! = 0$ tell the user their number is odd

The two separate parts a) and b) suggest using an if/else statement.

```c
#include <stdio.h>
#include <math.h>

int main(){
  //This is where we will store the user's input
  int N;

  //Ask the user for a number (this line prints to the screen
     only)
  printf("Please enter a number for checking\n");
  //This line actually waits for the number, and stores it in N
  scanf("%i", &N);

  //Check if N is even
  if (N % 2 == 0){
    //This block executes if  N % 2 == 0 is true
    // Note that the % in scanf / printf is NOT the math op mod
    printf("Your number %i is even\n", N);
  } else {
    //This block executes if N % 2 == 0 is false
    printf("Your number %i is odd\n", N);
  }

  return 0;
}
```

even.c

Please read through and understand what this code is doing – you will be making changes to this code to accomplish different tasks this morning.

Is the number N divisible by d?

Warm up
Make a copy of even.c, and call it mod3.c. Change mod3.c so it tells you whether or not the number you enter is divisible by three.

Divisibility
Make a copy of even.c, and call it divisible.c. Here is what we want the program to do:

1. Get a number N to check
2. Get a number d to divide by
3. Your program should tell you whether or not N is divisible by d (or if d is a factor of N)

A sample run might look something like this (user input in bold)

```
Please enter a number you wish to check for factors
21
Please enter a suspected factor
7
7 is a factor of 21
```

The same program taking different inputs from the user might look like the following

```
Please enter a number you wish to check for factors
21
Please enter a suspected factor
6
6 is not a factor of 21
```

Is the number N prime?

Counting factors
Make a program factor.c by either copying an existing program, or starting fresh. Your program's task is to count the number of factors of the inputed number. For example:

1. If $N = 21$ take your we would see that 21 can be divided by 1, 3, 7, and 21. Your program would return 4, as 21 is divisible by 4 numbers.

2. If N was 30, we would see that N can be divided by 1,2,3,5,6,10,15,30. Your program would return 8, as 30 is divisible by 8 numbers.

Here is a few hints:

1. Introduce a new variable factor that keeps track of how many factors of N you have found.

2. We know that the factors of N all lie between 1 and N (including 1 and N). Loop though each number from 1 to N, and if it is a factor add one to factor.

3. Once you have looped through all possibilities, report the number of factors.

An example of your program running would be

```
Please enter a number you wish to check for factors
21
21 has 4 factors
```

Extension
You should be able to alter your program slightly to check if the number N is prime. Recall a number is prime if its only factors are 1 and itself.

NIM – a simple game

In this program, our goal is to have a collection of sticks (say 9), and the rules are that each player may take 1, 2, or 3 sticks on their turn. The person that takes the last stick loses. Here would be a sample game:

Setup :		(9 sticks remaining)
Turn #1:	Player # 1 takes 2 sticks	(7 sticks remaining)
Turn #2:	Player # 2 takes 1 stick	(6 sticks remaining)
Turn #3:	Player # 1 takes 3 sticks	(3 sticks remaining)
Turn #4:	Player # 2 takes 2 sticks	(1 sticks remaining)
Turn #5:	Player # 1 takes 1 stick	(0 sticks remaining)

So in this game, Player #1 would lose.

We will write this program together as a class, but first we are going to spend some time thinking about it.

1. The first thing to do is get a "feeling" for the game. Play a game with each person sitting around you.

2. After playing the game a few times, think about what the computer is going to have to keep track of. Make a list and compare it to people around you.

3. Can you identify some of the things our program is going to need to do? How is it going to check whose turn it is? How will it check if the game is over? Come up with suggestions (you may find that your solutions need you to keep track of things you had not thought of in the previous step)

4. Look at your list of solutions and variables to keep track of, and compare with your neighbor. We are now in a good position to start programming!