PROJECT 1: THE EUCLIDEAN ALGORITHM

CISC 1100

Let $n, m \in \mathbb{Z}$; assume they are positive. (Minor changes can handle the case when n, m are negative but we won't concern ourselves with that.)

The divisors of an integer are the numbers which divide it with no remainder.

e.g.) The divisors of 12 are 1,2,3,4,6,12.

A common divisor between two integers n, m are all of the divisors which are shared.

e.g.) The divisors of 14 are 1,2,7,14; the divisors of 12 are 1,2,3,4,6,12. The common divisors of 14 and 12 are 1,2.

The greatest common divisor or GCD of two integers is, as you would imagine, the largest of the common divisors. This is notated either $gcd(n, m)$ or simply (n, m) when the context is clear.

e.g.) $(12, 14) = 2$

e.g.) $(52, 39) = 13$

Notice that if d is a divisor of n, then $d \leq n$.

e.g.) 1,2,7,14 are all less than or equal to 14.

Consequently, $gcd(n, m) \leq n$ and $gcd(n, m) \leq m$, since it is a divisor of both. This fact will be useful in a moment.

1. The basic algorithm

The Euclidean algorithm is a method of determining the greatest common divisors of two integers n, m. It works by considering remainders.

e.g.) Find $gcd(212, 36)$.

First, $212 \div 36 = 54R32$, i.e. $212 = 36 * 5 + 32$

Next, $36 \div 32 = 1R4$, i.e. $36 = 32 * 1 + 4$

$32 \div 4 = 8R0$, i.e. $32 = 4 * 8 + 0$

So, the greatest common divisor is 4!

Here's why it works:

Theorem Let n, m be given; we know that there is some $0 \leq r < m$ such that $n \equiv r (mod\, m)$. But this means there is some $q \in \mathbb{Z}$, $n = qm + r$.

Therefore, $gcd(n, m) = gcd(m, r)$.

PROOF: We will show that $gcd(n, m) \leq gcd(m, r)$ and $gcd(m, r) \leq gcd(n, m)$, which means that $gcd(n, m) = gcd(m, r)$.

First we notice that $n = qm + r$. Since $gcd(m, r)$ is a divisor of both m and r, it must be a divisor of n. Therefore $gcd(m.r)$ is a divisor of both n and r, so $gcd(m.r) \leq gcd(n.m)$ (it is a divisor and the largest divisor is the gcd).

Next notice that $r = n - qm$. Then since $gcd(n, m)$ is a divisor of both n, m, it must be a divisor of r as well. Therefore $gcd(n, m) \leq gcd(m, r)$. QED

What the algorithm is doing is to keep replacing the the divisors with remainders. Because of the fact given above, this process will keep giving the same GCD. It stops when the remainder is 0. We define $gcd(n, 0) = n$, which is entirely made up. (Basically, your answer is the last remainder before it became 0.)

2. Useful code

2.1. **Displaying information.** Javascript will not display everything that a program does. It will only display when it's told to. The command for this is "console.log()", and will display anything within the parentheses. Text must be within quotations marks "". Mixing text and variables is done via + signs, e.g. if $x = 4$ then

$$console.log(x + \text{" is a number."})$$

will come up as '4 is a number.'

2.2. **Declaring variables.** A variable is a form of data. For our purposes, we will only use variables which are numbers.

Variables can have any name that has no punctuation or spaces. For instance "x" can be a variable, and so can "hereIsAVariable".

At the beginning of your program, it's a good idea to declare all variables. This is done with the code "var x", where x is the name of the variable.

We can ask for the value of a variable with the code prompt. When run, this will make a pop-up window where we can type information.

For example:

$$var\, x = prompt(\text{"Enter a number"})$$

will cause a pop up to open, ask for a number, and then store that number as "x".

Variables can also be stored in terms of other variables. For example

$$var\, x = prompt(\text{"Enter a number"})$$

$$var\, y = x - 3$$

will prompt for x and store y as x-3.

2.3. **If, then.** We often make logical statements in the form "if, then." It would be helpful if a computer could listen to such instruction. For instance if a program is designed to warn people before they overdraft an account, we would want: If the account a balance is less than 100, then warn.

The code for this is of the form $if(){}$. The "()" contains the statement that must be true to run. The {} is the code that will be executed. Javascript doesn't need the words "then."

For example,

$$if(accountBalance < 100)\{console.log(\text{"You're running low on money!"})\}$$

However, there is one hang up. An "if" statement needs to tell the computer what to do if there is a false statement (e.g., what do I do when accountBalance$\geq$ 100?).

This is accomplished by adding the code "else{}". The {} is the code that will run if "()" is false.

E.g.,

$$if(accountBalance < 100)\{console.log(\text{"You're running low on money!"})\}else\{\}$$

will tell a computer to do nothing if accountBalance$\geq$ 100.

2.4. **Do loop.** What if we want to run a process a whole bunch of times? We can use do loops.

These work a lot like if statements. They look like "do{} while ()". Basically, the code comes after the "do", and the condition comes in the "()"–after the code, instead of before it.

The computer will repeat the code until the condition is met.

E.g.,

$$var\, counter = 10$$

$$do\{console.log(\text{"stop hitting yourself"})$$

$$counter = counter - 1\}while(console! == 0)$$

will make the console log "stop hitting yourself" ten times.

3. Programming the basic algorithm

Let's learn some functions for JavaScript.

"Math.floor()" will return the whole number component of whatever comes inside the parentheses.

e.g.) Math.floor(15)=15

e.g.) Math.floor(13.768)=13

e.g.) Math.floor(5/3)=1

Question 1 Suppose that $x = Math.floor(n/m)$. Explain why x is the quotient when n is divided by m, and $n - mx$ is the remainder.

Question 2 Let $y = n - mx$. What does y mean?

Let's look at what happens with some actual numbers. Let $m = 4, n = 18$. Then $x = Math.floor(17/4) = 4$ and $y = 2 = 18 - 4 * 4$.

Question 3 Suppose we repeat the process, but this time use $x' = Math.floor(m/y), y' = m - xx'$. What does this do?

Let's look at our example again. We found $m = 4$ and $y = 2$. So, $x' = Math.floor(4/2) = 2$ and $y' = 4 - 2 * 2 = 0$.

Question 4 Imagine we want to write a program which continually replaces x', y' as we did above. We want the process to stop when $y' = 0$. What should our "while" condition be?

PROJECT: Write code which will display all steps of the Euclidean algorithm. This code should:

1) Ask for two integers $n =, m =$. Use "prompt()".

2) Compute and display $x = Math.floor(n/m), y = n - mx$. Use "console.log()".

3) Replace the divisors with the remainders. Use "do{} while()".

4) Terminate when $y = 0$ and display "The greatest common divisor of n, m is [the answer]."

Copy and paste a picture of just your code. Then, run the program for:

1) $n = 60, m = 15$.

2) $n = 225, m = 115$.

3) $n = 440, m = 309$.

4) $n = 191, m = 74$.

For each of these, copy and paste a picture of the console once the program has run.