

Program #3 - Padovan ASM

Print the Padovan sequence with an Intel assembly language program using gcc

- Due: **Fri Feb 28, 2014**
- Worth: **15 points**

Good luck!

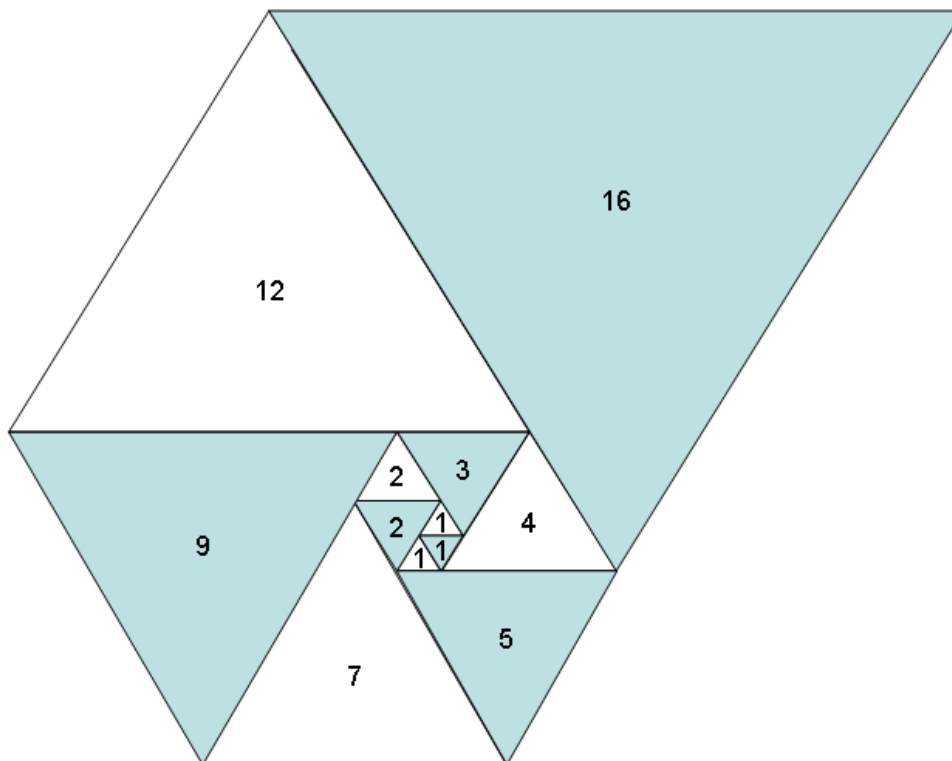
1. Description

Please write an Intel assembly program that prints the Padovan sequence. Read on!

🔗 Padovan Sequence

The Padovan sequence is a sequence of integers defined by a recurrence relation. You can read a summary here: en.wikipedia.org/wiki/Padovan_sequence.

You can visualize the Padovan sequence $\{1, 1, 1, 2, 2, 3, 4, 5, 7, 9, 12, 16, \dots\}$ as a (totally cool) spiral of triangles.



Can you draw in the next triangle with side of length 21? Then 28? Then 37?

The Padovan sequence is defined by the following recurrence relation:

The **Padovan sequence** is the [sequence of integers](#) $P(n)$ defined by the initial values

$$P(1) = P(2) = P(3) = 1,$$

and the [recurrence relation](#)

$$P(n) = P(n - 2) + P(n - 3).$$

The first few values of $P(n)$ are

1, 1, 1, 2, 2, 3, 4, 5, 7, 9, 12, 16, 21, 28, 37, 49, 65, 86, 114, 151, 200, 265, ... (sequence [A000931](#) in [OEIS](#))

Padovan also has a definition for negative indices.

Extension to negative parameters [\[edit\]](#)

As with any sequence defined by a recurrence relation, Padovan numbers $P(m)$ for $m < 0$ can be defined by rewriting the recurrence relation as

$$P(m) = P(m + 3) - P(m + 1),$$

Starting with $m = -1$ and working backwards, we extend $P(m)$ to negative indices:

P_{-20}	P_{-19}	P_{-18}	P_{-17}	P_{-16}	P_{-15}	P_{-14}	P_{-13}	P_{-12}	P_{-11}	P_{-10}	P_{-9}	P_{-8}	P_{-7}	P_{-6}	P_{-5}	P_{-4}	P_{-3}	P_{-2}	P_{-1}	P_0	P_1	P_2
7	-7	4	0	-3	4	-3	1	1	-2	2	-1	0	1	-1	1	0	0	1	0	1	1	1

Your program

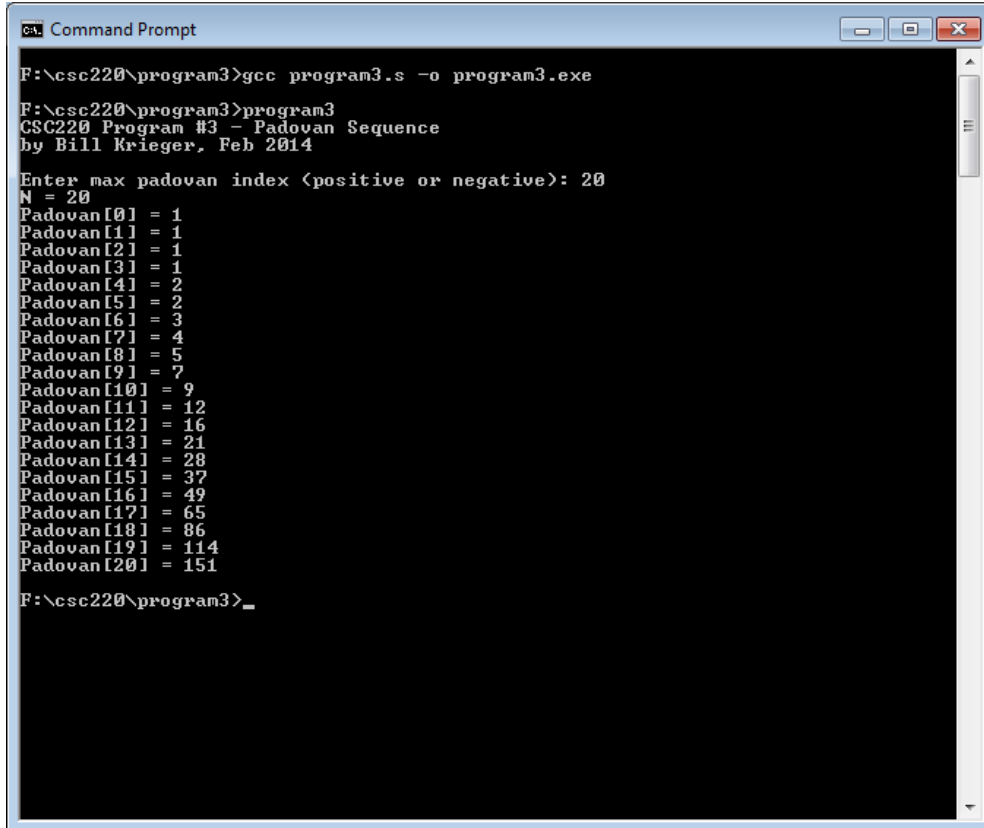
Please write your program in Intel assembly using gcc. We will be discussing this at some length in class.

Some more details:

- Separate your solution into 2 Padovan functions: one for positive indices and a separate one with negative indices.
- Please implement your positive Padovan function using **recursion**.
- Please solve your negative Padovan sequence using **iteration**.
- You **must** liberally comment your code. Describe each logical block! This is important for two reasons: 1) it allows you to keep track of what your code is doing, and 2) it shows me that you understand the code you have written.
- You can use one or multiple files for your program. It's up to you.

My program

Your output should look something like this. In the PC console window below, I show 2 things: First, I use the gcc command to assemble and link my program. Then, I run it for a positive Padovan index of 20.

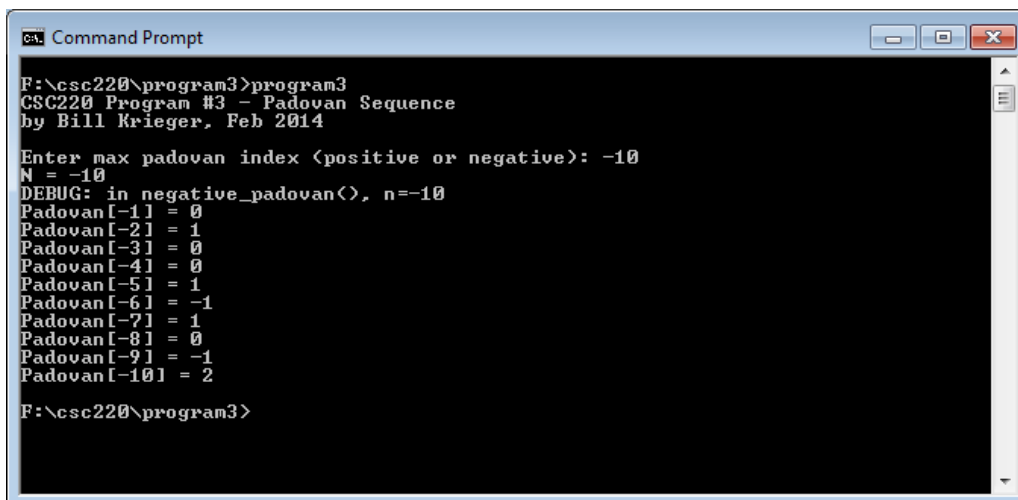


```
Command Prompt
F:\csc220\program3>gcc program3.s -o program3.exe
F:\csc220\program3>program3
CSC220 Program #3 - Padovan Sequence
by Bill Krieger, Feb 2014

Enter max padovan index (positive or negative): 20
N = 20
Padovan[0] = 1
Padovan[1] = 1
Padovan[2] = 1
Padovan[3] = 1
Padovan[4] = 2
Padovan[5] = 2
Padovan[6] = 3
Padovan[7] = 4
Padovan[8] = 5
Padovan[9] = 7
Padovan[10] = 9
Padovan[11] = 12
Padovan[12] = 16
Padovan[13] = 21
Padovan[14] = 28
Padovan[15] = 37
Padovan[16] = 49
Padovan[17] = 65
Padovan[18] = 86
Padovan[19] = 114
Padovan[20] = 151

F:\csc220\program3>_
```

In my second run, I'll use a negative index.



```
Command Prompt
F:\csc220\program3>program3
CSC220 Program #3 - Padovan Sequence
by Bill Krieger, Feb 2014

Enter max padovan index (positive or negative): -10
N = -10
DEBUG: in negative_padovan(), n=-10
Padovan[-1] = 0
Padovan[-2] = 1
Padovan[-3] = 0
Padovan[-4] = 0
Padovan[-5] = 1
Padovan[-6] = -1
Padovan[-7] = 1
Padovan[-8] = 0
Padovan[-9] = -1
Padovan[-10] = 2

F:\csc220\program3>
```

2. Grading

Please create a **program3** folder in your k: drive space. I'll look for these files:

- Your **README.txt** file where you describe the state of your program
- Your assembly language program. Please put `main()` in **program3.s**. You can place your other functions in this file or create others as you like.
- Once your code is running try a number of positive and negative max index values. Describe these tests in your README.
- I will **not** ding your grade if you don't check for **overflow**. I didn't.

3. Notes

Here's some mighty wind to fill the sails of your Program #3 expedition.

Your process

I *strongly* recommend you take the following steps en route to Program #3 nerdvana:

- Do your program in Java or C first. This should be easy.
- With that complete, print your code. Circle a tiny sliver of your program and code it up in assembly. Test and run it. Once one sliver is done, then move on to the next. If you get stuck on a sliver, then ask for help. Rinse and repeat.
- Start a function with my template assembly code in the k: drive.
- I used Notepad++ to edit my files. I ran gcc in the Windows console.

A cautionary note - plagiarism

It's GREAT to get help from me: via email or in person. It's OK to talk to your peers as well.

But you know you've crossed the line and cheated when:

- You copy-paste code from someone else
- You don't understand all your code
- You change variables from someone else's existing code

Start early. Ask for help if you need it.

Thanks, Bill