What should I do with the following?

```
0101111010101000010101000011110111111001001100101010110001000001100011110111
1101101111000011010011101100110010101011000100000110000111011101100100010000
1100101111100011001011100011111010111101001100101010000011101001101000110010
0100000110011011011111101100110110011011111110111110100111011101100111010000
1100011110111111001001100101010000011011111101110010000011000010100000111011
1100001111001011010011100101111010011110010100000110111111001100100000110111
1110000110010111100101100001111010011010011101110110011101000001110011111100
1110011111010011001011101101111001101000000101000110000111101000100000111010
1101000110010101000001110110110010111100101111001010000011011001100101110000
1110011111010001000001110100111001011110010100000110000101000001110110110010
1110010111001111010011101111110111001000001101111110011001000001001100110100
1101110111010111110000100000110000111011101100100010000011000010100000111011
1100101111001011100111101001110111111011100100000110111111001100100000101011
1101001110111011001001101111111011111100110101001010111001000000100000100001
1101111110110111011011100101110111011101000100000110111111011100100000111100
1101111111010111100100100000110111111000101110011110010111100101110110110000
1110100110100111011111101110111001101011100100000010000010101001101000110010
1101110010000011000111101111111011011101101110010111011101110100010000011011
1101110010000011101111101000110000111101000100000111100111011111110101010000
1110100110100011010011101110110101101000001110100110100011001010100000110001
1101111110010011001010100000110100111100110101100010000011101111110100011000
1110100010000011010011110100010000011001001101111110010111100110101100010000
1101000110111111101110100000110000111011100100000110000111101001110100110000
1100011110101111001011110010010000011011011101001110011111010001110100010000
1110101111001111001010100000110100111101000101100010000011000011101110110010
0100000111011111010001100001111010001000001111001110111111101010100000110110
1101001110011111010001110100010000011001001101111010000011101001101111010000
1100100110010111000011101100010000011101111101001111010011010000100000111001
1110101110001111010000100000110000111011100100000110000111101001110100110000
1100011110101101011100001010010101001011110001010000101011010011101110111010
0100000110110111000011101001110111001010001101001110111011101000100000110000
1110010110011111000110101100010000011000111101000110000111100100101010010101
0100000110000111100101100111111011001010010001010111101100010100001001110011
1101111111001001000000101000011101101110110101001000101000010010001001111001
1111001111001111101001100101110110101010001100001111001011001111110110101101
0110000101110101010010111011000101011111101
```

I played around with this and overthought it plenty. Eventually I arrived at the conclusion that this is a binary representation of an ASCII string because running the following in python 2 (with the above stored as a string called bstring)

```
    print len(bstring) % 7
```

outputs 0.

So I completed the script to split the bit string into segments 7 digits long, interpret the integer in base 2, then get the ASCII character associated with that value and append it to a new string.

This is the script:

```
bstring = "0101111...111101"

newstr = ""
for li in xrange(0, len(bstring), 7):

        byte = bstring[li:li+7]
        value = int(byte, base=2)
        char = chr(value)
        newstr += char

print newstr
```

The result is this:

```
/*
Code, compile, and execute the following code on a variety of operating
systems (at the very least try a version of Linux and a version of Windows).
Comment on your observations.  Then comment on what you think the code is,
what it does, how an attacker might use it, and what you might do to deal
with such an attack.
*/

int main(int argc, char** argv)
{
        for (;;)
                system(argv[0]);
}
```

Looking at this I immediately recognize a fork bomb because I Googled fork bomb implementations in a variety of languages a few months ago (thanks Wikipedia). I wrote one in Windows 7 batch because I wanted to see how many instances of CMD it would take to crash the VDIs they give us at Southwest Airlines.

Basically the first argument passed into argv will always be the name of the executable, so calling system with that string will just spin up another instance of the same process.
for (;;) is the same as while (true).

Anyway:
Because I am a good boy I compiled and executed this on Ubuntu Linux (WSL) and Windows 10 Home edition (with no protection because I want a reason to install arch, unfortunately everything was fine) with GCC (MinGW on Windows).

Ubuntu Linux:
Because I used WSL for this, there was no GUI so it was pretty uneventful but still a learning experience.
The command I used (sol.py is the python file name)

```
python sol.py | gcc -xc - ; ./a.out
```

After pressing enter, I got a compiler warning because there are no #include’s for system(), but otherwise the command line went dead. Sending SIGINT did nothing. I gave it a minute to see how Windows would handle a fork bomb in WSL, but nothing happened so I closed the process and moved on. How boring!

Windows 10 Home Edition

I ran this directly on the hardware half expecting corruption to occur as a result of this irresponsible behavior (Unity Engine did it to me so why wouldn’t a fork bomb).

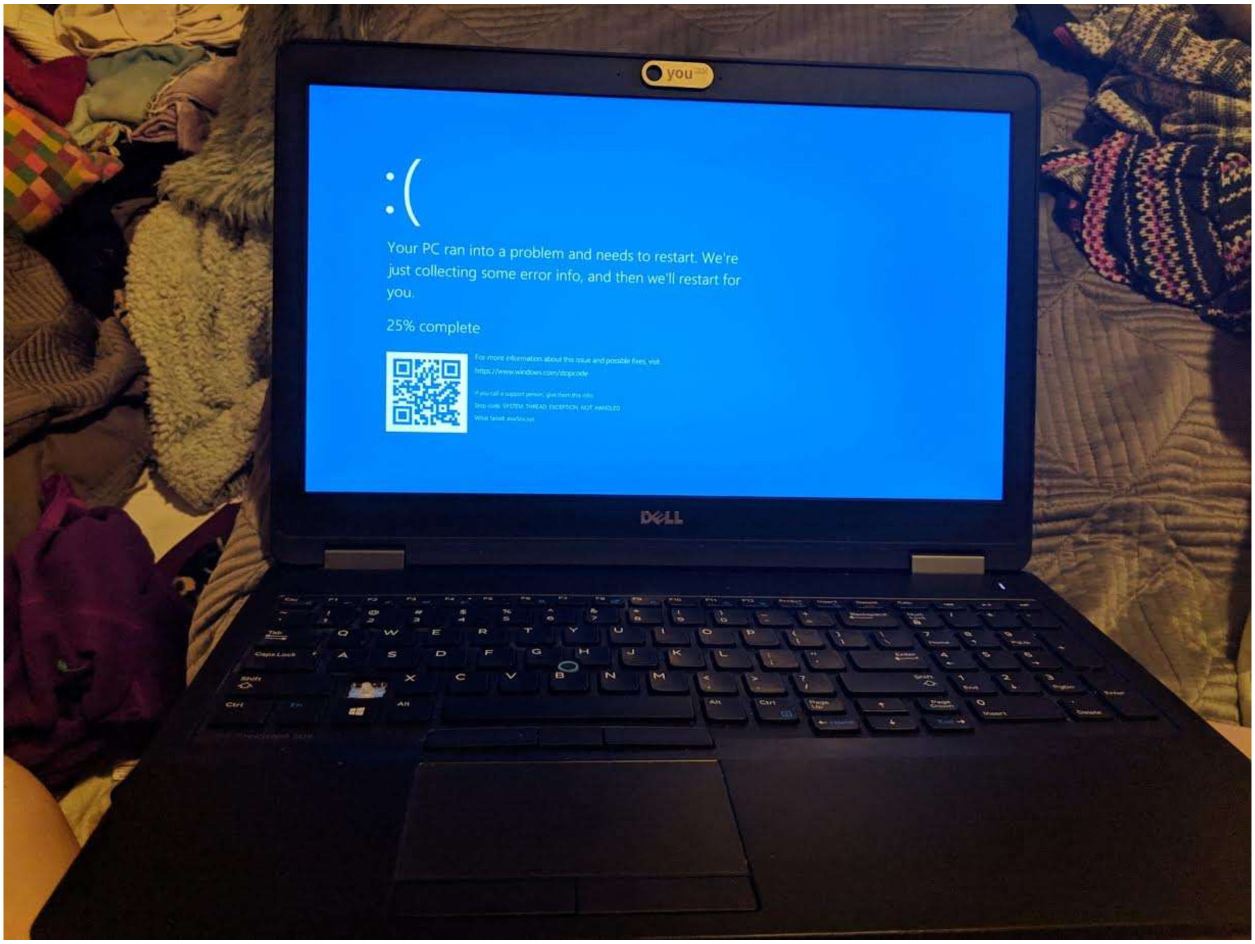
The command is pretty much the exact same (using the exact same file)

```
py2 sol.py | gcc -xc - && .\a.exe
```

py2 is an alias for my python 2 path because I use python3 by default.
For some reason MinGW gcc didn’t yell at me for not having the #include for system().

Anyway, this one was more fun. My computer got REALLY hot in like 10 seconds. My antivirus struggled to “scan” all of the processes being started, how could it compete? I’m not sure why I keep it around anyway. Then I got a 5 second freeze and a blue screen because Windows ran out of thread space :D.

Here is a picture to use in your future anti-microsoft rants.



:(

Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you.

25% complete

For more information about this issue and possible fixes, visit https://www.windows.com/stopcode

If you call a support person, give them this info:

Stop code: SYSTEM THREAD EXCEPTION NOT HANDLED

What failed: aswSnx.sys

Continuing to follow the instructions in the C comment:

The advantage of this kind of attack would be that the system effected loses all of its compute resources to trying to process the attack, so the user you are trying to troll can end up with an unresponsive computer until their system crashes. In terms of cyberstorm, this means they are probably not earning points. In terms of real world applications, if you're trying to down somebody's website (etc) in a non DDoS way this might work assuming you can get access to whatever box(es) it runs on. It may be worth tweaking the fork bomb such that you keep the system at processing capacity but have it self-limit so that it doesn't crash the system.

Google says a good way to prevent a fork bomb (on Linux) is to set a process count limit on a user or group in /etc/security/limits.conf.

https://www.cyberciti.biz/tips/linux-limiting-user-process.html

They didn't like my adblock.

My own idea to prevent this specific fork bomb implementation would be to scan new processes for duplicate names, because each process instance will have the same name. Say if 30 instances of 'nonsuspiciousprocessignoreme' spin up in like 50 ms then obviously I will probably want to stop that.