CS 5 Review

Exam is ***Wed 5/12 @ 2 PM***

Only have 5 minutes? Life+C4

May bring 2 pages of your own notes

Comprehensive… *something about everything!*

Except..

Connect-Four AI

vPython and other topic-specific libraries

Picobot, *alas*!

Turing machines

Different-sized infinities!

of strings

# of Python programs

# of int ⇒ bool functions

# of Booleans

# of integers

# of real numbers

Not infinite at all

Countably infinite

Uncountably infinite (whew!)

State machines

Create a state machine that accepts strings with ***at most*** two identical consecutive bits.

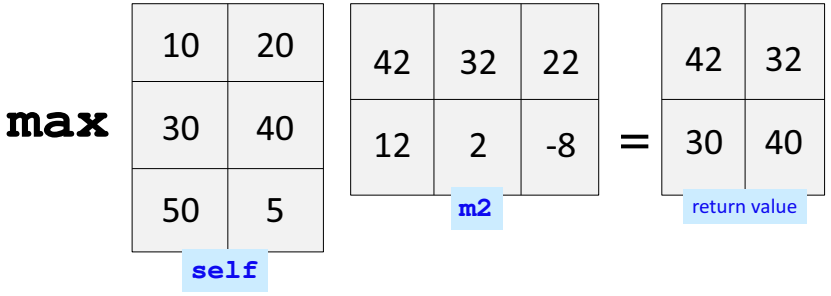
…in as few states as possible …

Classes, objects, and 2D data

```
class Matrix:
    def __init__(self, height, width):
        self.H = height
        self.W = width
        self.data = [[0]*self.W for row in range(self.H)]
```

```
def max(self, m2):
```

Here's the picture:



What are the NEW height and width?

Classes, objects, and 2D data

```
class Matrix:
    def __init__(self, height, width):
        self.H = height
        self.W = width
        self.data = [[0]*self.W for row in range(self.H)]

    def max(self, m2):
```

Here's the picture:

2D data

```
def symmetric(S):
    """Returns True if S is symmetric around the NW-SE line.
       returns False otherwise."""
    H = len(S)
    W = len(S[0])

    for r in range(H):
      for c in range(W):

      if                                                    :
          return False
```

1	2	3
2	4	5
3	5	6

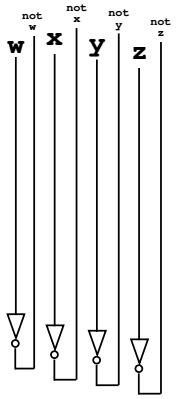
```
S = [[1,2,3],
     [2,4,5],
     [3,5,6]]
```

How could we compare corresponding elements ***across the diagonal***?!

Where should we return True?

Primality-testing circuit!

	w	x	y	z	isPrime?
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	
9	1	0	0	1	
10	1	0	1	0	
11	1	0	1	1	
12	1	1	0	0	
13	1	1	0	1	
14	1	1	1	0	
15	1	1	1	1	



Assembly

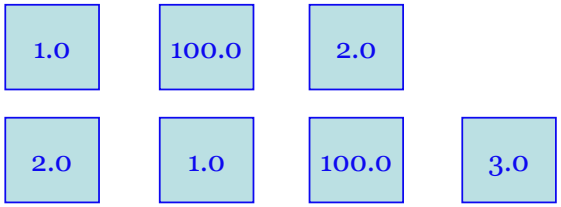
```
0  read r1          # r1 is our input, assumed > 0
1  setn r9 0        # r9 = 0      r9 is the "answer"
2  copy r2 r1       # r2 = r1     r2 is our "loop index"
3  nop
4  nop
5  nop
6  jeqz r2 14       # While r2 != 0
7  div r3 r1 r2     # r3 = r1/r2; r3 is a "scratch pad"
8  mul r3 r2 r3     # r3 = r2*r3
9  sub r3 r1 r3     # r3 = r1-r3
                    # r3 is the remainder: r3 = r1%r2
10 jgtz r3 12       # If r3 > 0, we skip the next line
11 addn r9 1        # If we're here, r1%r2 == 0, meaning that
                    # r2 divides r1, so we add 1 to r9
                    #
12 addn r2 -1       # r2 -= 1
13 jumpn 6          # Continue the while r2 != 0 loop…
14 write r9         # When finished, we print r9, which
                    # ..represents the number of divisors
15 halt
```

All Hmmm instructions will be provided…

r9	r2	r1
answer	loop index	input

Loops and recursion...

```
def median(L):
    """Returns L's median--the middle element OR
       the average of the two middle elements."""
```



Loops...

`isPrime(12) ⇒ False` `isPrime(11) ⇒ True`

```
def isPrime(n):
    """Check all possible factors (pf).
       If a pf divides n, return False.
       Otherwise, return True."""
    for pf in range(2, n):

        if                          :
```

`addPrimes([5, 6, 7]) ⇒ 12` `addPrimes([42, 44]) ⇒ 0`

Recursion...

```
def addPrimes(L):
    """Sums all and only the primes in L."""
    if L ==

    elif

    else:
```

Base case?

prime?

or not...

...or list comprehensions!

Recursion...

```
def fib(n):
    if n < 2:
        return 1
    else:
        return fib(n - 1) + fib(n - 2)
```

How many times is fib called in **fib(5)** ?

Recursion...

```
def uniquify(L):
    """Returns L's distinct elements--no repeats!"""
    if   ________________:
    elif ________________:
    else:
```

Base case?

One side...

The other...