Week 6 Exercises
Copeland Carter

1. OH!! SPACE!! I LIKE SPACE!!
Much more than the Command Module. Because it's changing 105 m/s VERY quickly, because it has a lot less mass.
Though, I suppose the *magnitude* of acceleration is the same, because Newton's 3rd law and all that, it just affects the Command Module much less because it has less mass. So actually c.
ANSWER: c

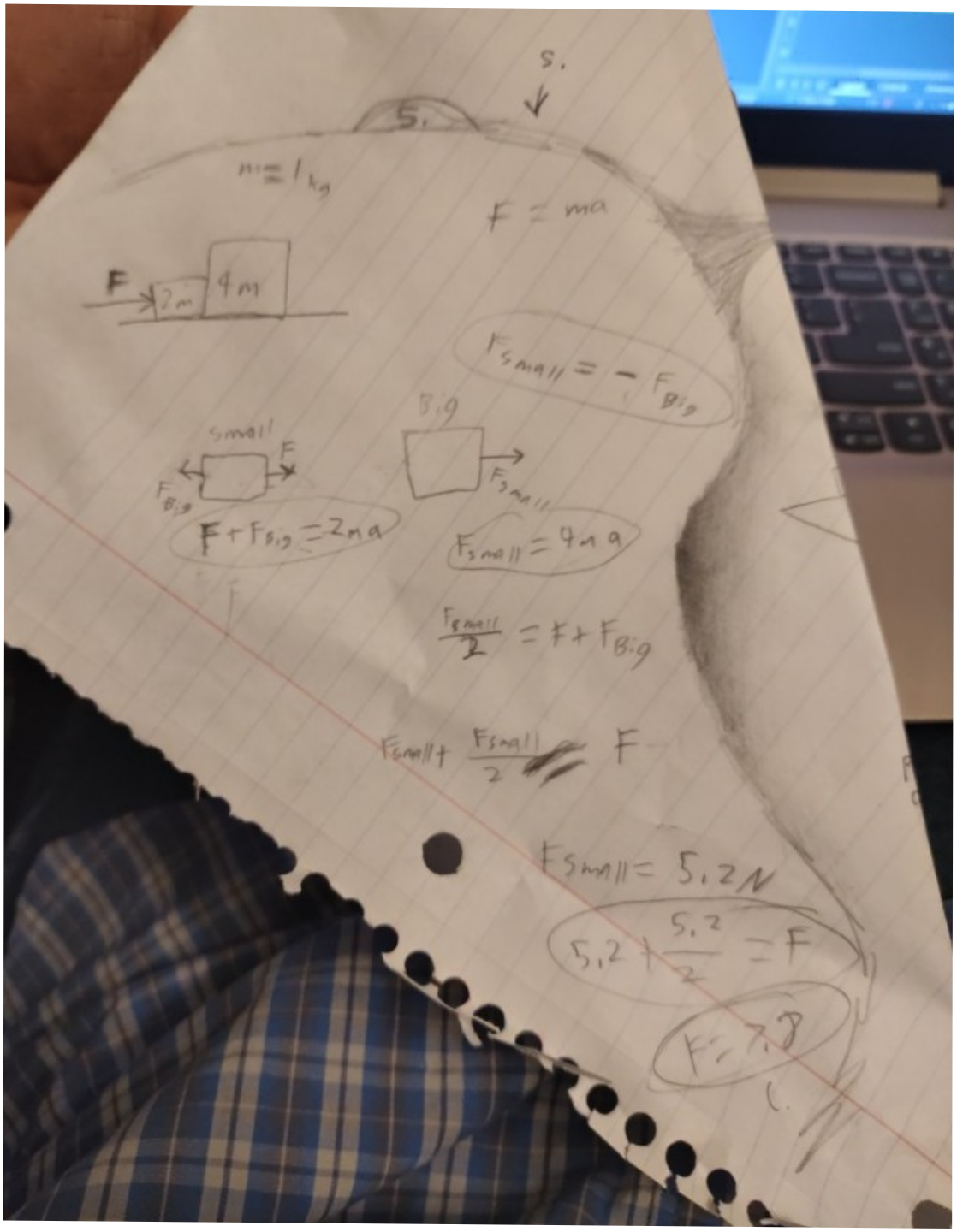
2. Ahh, I get it now. It's not a typo. Smart.
So same answer. The velocities are irrelevant, they're hitting, which means they have an equal force.
ANSWER: c

3. Well seeing as how I STILL HAVE NO IDEA WHAT b IS...
grumble grumble namespace grumble single letter variables grumble math grumble grumble
well lets see...
newtonsLaw(m=3*(1/1000), a=9.8)
    -> 0.0294
and that's force in Newtons, so then...
Nope, still don't actually now how to solve this problem. Do I know R?
[solve(Eq(var('R'), -b*2), R) for b in (.1, 1, 1.5, 2.3, 4.5)]
    -> [-0.200000000000000, -2, -3.00000000000000, -4.60000000000000, -9.00000000000000]
So R is one of those... a almost makes sense, if you squint....
is R Newtons / something?
I don't know. I'll come back to this questions.
ANSWER: COME BACK TO ME

4. ummmm....
M=2
newtonsLaw(m=2*M, f=12)
    -> 3.0
SHOW WORK LATER
ANSWER: b

5.
ANSWER: c

My work for 5 -->



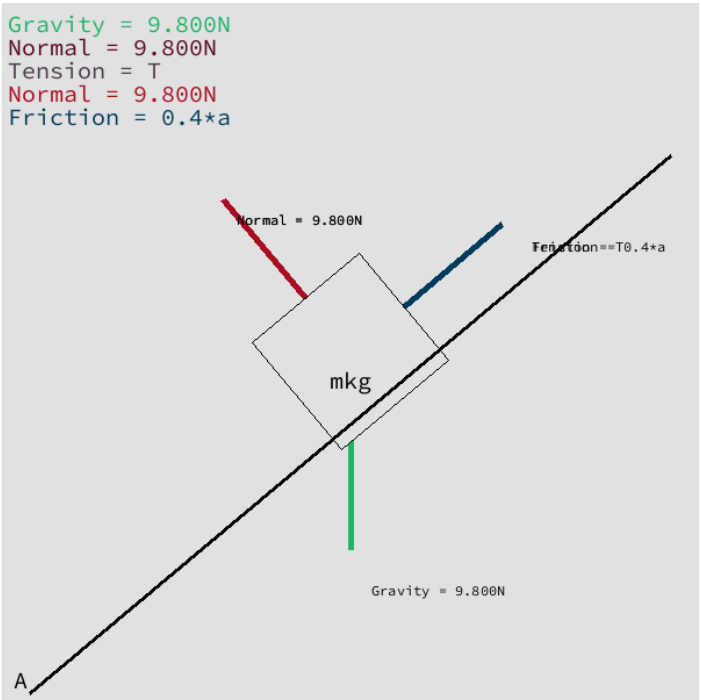
6. This is as far as I got... I think I maybe need to add matricies?

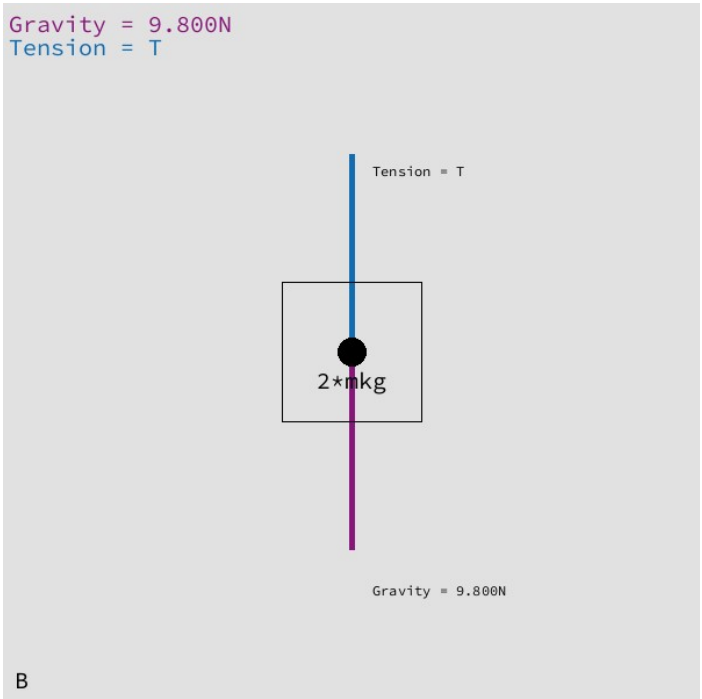
```
A = Particle2D(var('m'))
B = Particle2D(2*m, includeNormal=False)

A.addForce(Vector2D(var('T'), 40, False), 'Tension')
A.addForce(Vector2D(9.8, 40+90, False), 'Normal')
A.addForce(Vector2D(var('a') * .4, 40, False), "Friction")

B.addForce(Vector2D(T, UP), "Tension")

AccelInTermsOfTension = solve(Eq(A.netForce().r, A.mass * a), a)[0]
TensionInTermsOfAccel = solve(Eq(B.netForce().r, B.mass * a), a)[0]
# AccelInTermsOfTension.subs(a, TensionInTermsOfAccel)
A.diagram(); B.diagram()
```



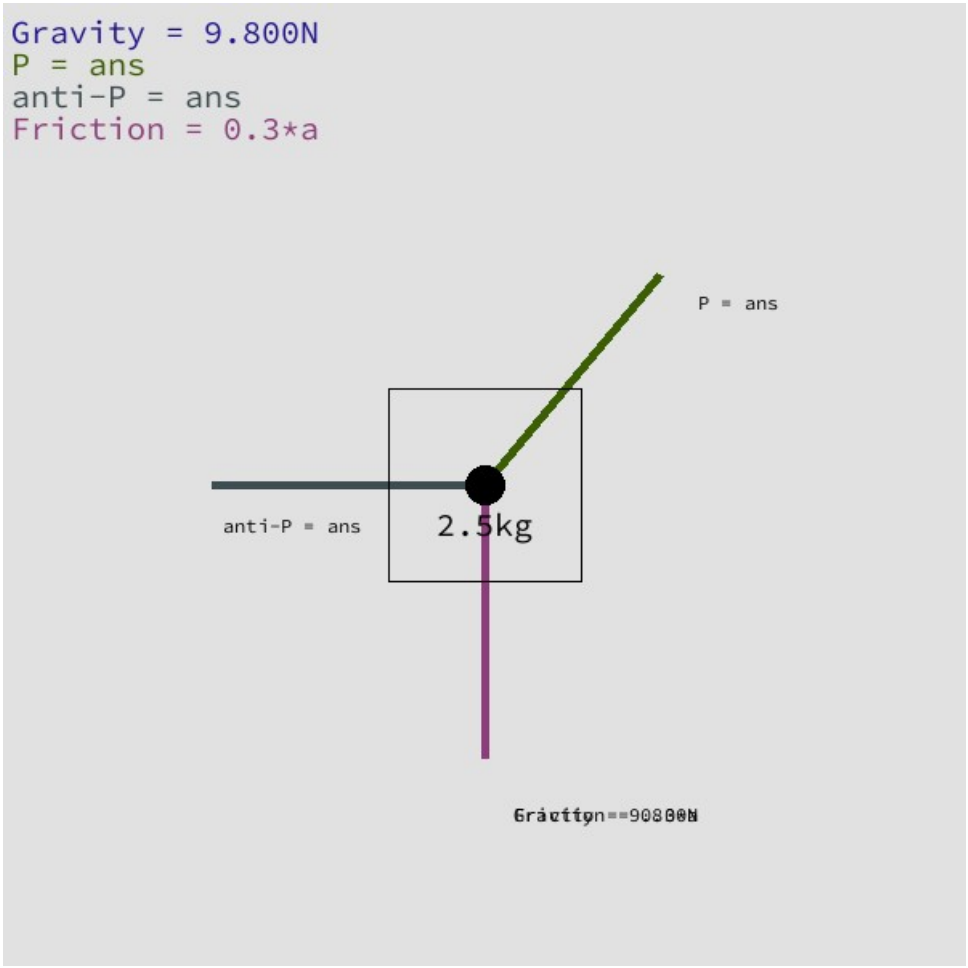


8. (Yes, I skipped one)
Here's what I got: I'm not sure if I'm using friction correctly…
(I'm also not sure why it stopped evaluating it all symbolically, but that's a code issue. Besides, it's just a giant mess of sin's, cos's, and asin's otherwise.)

```
block = Particle2D(2.5, includeNormal=False)
P = Vector2D(var('ans'), 50, False)
block.addForce(P, 'P')
# Would the P normal force be out, or opposite the wall?
# block.addForce(-P, 'anti-P')
block.addForce(Vector2D(P.r, LEFT), 'anti-P')
block.addForce(Vector2D(var('a') * .3, DOWN), "Friction")
print(solve(Eq(block.netForce().r, 0), ans))
```

→ [-6.92964645562817*sqrt(-0.00093710953769262 8*a**2 - 1),
6.92964645562817*sqrt(-0.00093710953769262 8*a**2 - 1)]

```
block.diagram()
```



10. (yes, I skipped another one)
I just realized I've been using 9.8 of gravity as force, not acceleration, which is where I went wrong in a lot of the previous questions.
But I'm not gonna go back and fix them, cause I'm out of time.
And I didn't procrastinate this week, either! I got it done early!
Humph.

But now that I realized that, I got this one down easy! It's just… something… times something else. I forgot already. Here's the code:

```
washer = Particle2D(75 + 12, includeNormal=False, gravity=Vector2D(9.8+.25, DOWN))
pull = Vector2D(var('ans'), UP) # Because the pulley is re-directing it
washer.addForce(pull, 'Pull')
washer.diagram()
```

ANSWER: a