Part B - Problem 4: TreatedPatient Class

(10/10 points)

We also need a representation for a patient that accounts for the use of drug treatments and manages a collection of `ResistantVirus` instances. For this, we introduce the `TreatedPatient` class, which is a subclass of `Patient`. `TreatedPatient` must make use of the new methods in `ResistantVirus` and maintain the list of drugs that are administered to the patient.

Drugs are given to the patient using the TreatedPatient class's `addPrescription()` method. What happens when a drug is introduced? The drugs we consider **do not directly kill virus particles lacking resistance to the drug**, but prevent those virus particles from reproducing (much like actual drugs used to treat HIV). Virus particles with resistance to the drug continue to reproduce normally. Implement the `TreatedPatient` class.

Hint: `reproduce` function child resistances

If you are really unsure about how to think about what each child resistances should be changed to, here is a different approach. If the probability `mutProb` is successful, the child resistance switches. Otherwise, the child resistance stays the same as the parent resistance.

Test: TreatedPatient 1

`Create a TreatedPatient with virus that is never cleared and always reproduces.`

Output:

```
virus = ResistantVirus(1.0, 0.0, {}, 0.0)
patient = TreatedPatient([virus], 100)
Updating patient for 100 time steps
Test completed.
```

Test: TreatedPatient 2

`Create a TreatedPatient with virus that is always cleared and always reproduces.`

Output:

```
virus = ResistantVirus(1.0, 1.0, {}, 0.0)
patient = TreatedPatient([virus], 100)
Updating patient for 100 time steps
Test completed.
```

Test: TreatedPatient 3

`Test for adding duplicate prescriptions in TreatedPatient`

Output:

```
Test completed.
```

Test: TreatedPatient 4

```
Test addPrescription and getPrescription in TreatedPatient.
```

Output:

```
patient = TreatedPatient([], 100)
Adding prescription Drug I
Drug I in plist: True
Adding prescription Drug N
Drug N in plist: True
Adding prescription Drug V
Drug V in plist: True
Adding prescription Drug P
Drug P in plist: True
Adding prescription Drug O
Drug O in plist: True
Adding prescription Drug L
Drug L in plist: True
Adding prescription Drug T
Drug T in plist: True
Adding prescription Drug A
Drug A in plist: True
Adding prescription Drug D
Drug D in plist: True
Adding prescription Drug W
Drug W in plist: True
Adding prescription Drug I
Drug I in plist: True
len(patient.getPrescriptions()): 10
Adding prescription Drug N
Drug N in plist: True
len(patient.getPrescriptions()): 10
Adding prescription Drug V
Drug V in plist: True
len(patient.getPrescriptions()): 10
Adding prescription Drug P
Drug P in plist: True
len(patient.getPrescriptions()): 10
Adding prescription Drug O
Drug O in plist: True
len(patient.getPrescriptions()): 10
Test completed.
```

Test: TreatedPatient 5

```
Test of getting TreatedPatient's resistant pop
```

Output:

```
    virus1 = ResistantVirus(1.0, 0.0, {"drug1": True}, 0.0)
    virus2 = ResistantVirus(1.0, 0.0, {"drug1": False, "drug2": True}, 0.0)
    virus3 = ResistantVirus(1.0, 0.0, {"drug1": True, "drug2": True}, 0.0)
    patient = sm.TreatedPatient([virus1, virus2, virus3], 100)
    patient.getResistPop(['drug1']): 2
    patient.getResistPop(['drug2']): 2
    patient.getResistPop(['drug1','drug2']): 1
    patient.getResistPop(['drug3']): 0
    patient.getResistPop(['drug1', 'drug3']): 0
    patient.getResistPop(['drug1','drug2', 'drug3']): 0
    Test completed.
```

Test: TreatedPatient 6

```
Test for virus populations in TreatedPatient.
```

Output:

```
    virus1 = ResistantVirus(1.0, 0.0, {"drug1": True}, 0.0)
    virus2 = ResistantVirus(1.0, 0.0, {"drug1": False}, 0.0)
    patient = TreatedPatient([virus1, virus2], 1000000)
    patient.addPrescription("drug1")
    Updating patient 5 times
    Expect resistant population to be 2^5 +/- 10
    Expect total population to be the resistant population plus 1
    Test completed.
```