6.01 Final Exam: Fall 2009

Name: ANSWERS

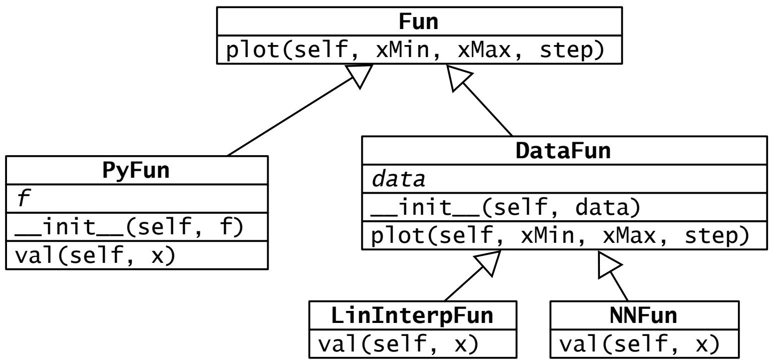
Enter all answers in the boxes provided.
You may use any paper materials you have brought.
You may use a calculator.
No computers, cell phones, or music players.

For staff use:

1.	/20
2.	/10
3.	/20
4.	/20
5.	/10
6.	/20
total:	/100

1 Object-Oriented Programming (20 points)

We will build a set of classes for representing functions (that is, mathematical functions, mapping real numbers to real numbers) in Python. Here is a diagram of the class hierarchy. Each box corresponds to a class, with its name shown in bold. Class attributes are shown in italics and methods are shown in regular text with their arguments. Arrows point upward to the superclass of a class. Fun has no superclass.



Every instance of a subclass of Fun that we create will have two methods: `val`, which takes an x value and returns the associated y value; and `plot`, which takes a description of the minimum and maximum x values for plotting, and the spacing between plotted points, and makes a plot. Different functions will be represented internally in different ways. In this problem, we will implement the `Fun`, `PyFun`, `DataFun`, and `LinInterpFun` classes; the `NNFun` class is in the diagram to illustrate where we might put a subclass of `DataFun` that uses a different interpolation strategy.

A. The class Fun is an *abstract superclass*. It won't be useful to make an instance of it, but we can put the `plot` method, which is shared among its subclasses, in it. The `plot` method should create two lists:

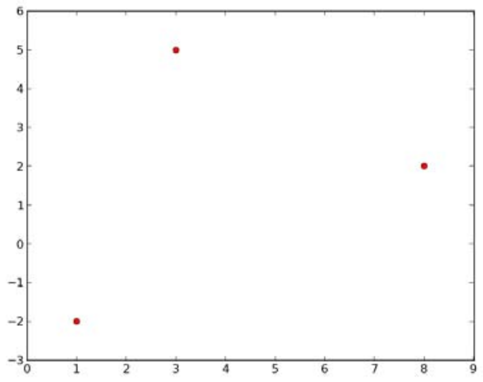
- `xVals` is a list whose first value is `xMin`, and whose subsequent values are `xMin + step`, `xMin + step + step`, etc., stopping so that the last value is as large as it can be, but less than `xMax`. `step` will typically be a float.
- `yVals` is a list of the y values of the function, corresponding to the x values in `xVals`. You can use the `val(self, x)` method, which is required to be defined for any actual instance of a subclass of `Fun`.

Then, it can call the global function `makePlot` on those lists. Assume `makePlot` is already defined.

So, for example,

```
makePlot([1, 3, 8], [-2, 5, 2])
```

would make the plot on the right.

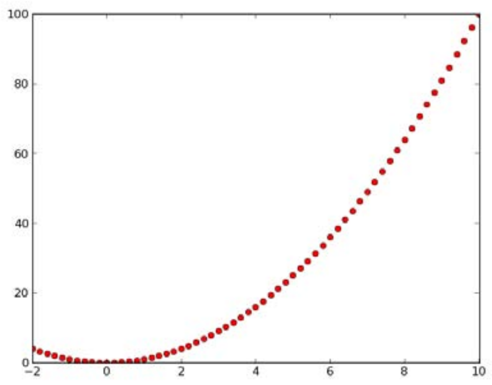


Implement the `plot` method. Recall that Python's `range` only works with integer step size.

```
class Fun:
    def plot(self, xMin, xMax, step):


        xVals = []
        yVals = []
        x = xMin
        while x <= xMax:
            xVals.append(x)
            yVals.append(self.val(x))
            x += step
        makePlot(xVals, yVals)
```

B. The PyFun class is a subclass of Fun. It represents a function just using a Python procedure of a single argument. It should operate as follows:

```
>>> t1 = PyFun(lambda x: x**2)
>>> t1.val(2)
4
>>> t1.val(-3)
9
>>> t1.plot(-2, 10, 0.2)
```



Implement the PyFun class.

```
class PyFun(Fun):
    def __init__(self, f):
        self.f = f
    def val(self, x):
        return self.f(x)
```

C. The `DataFun` class is a subclass of `Fun`, representing functions using a list of (x, y) data points, and using an interpolation rule to evaluate the function at values of x that don't appear in the data set. The data points are not sorted in any particular way. Different subclasses of `DataFun` will provide different interpolation strategies by implementing different `val` methods. `DataFun` is an abstract class, because it does not itself provide a `val` method. It will provide useful `__init__` and `plot` methods, however.

- The `__init__` method should take a list of (x, y) data pairs as input and store it in an attribute called `data`.
- The `plot` method should first plot the function with regularly-spaced x values, using the `plot` method from the parent class, `Fun`; then it should plot the actual data points stored in the `data` attribute.

Implement the `DataFun` class.

```
class DataFun(Fun):
    def __init__(self, data):
        self.data = data
    def plot(self, xMin, xMax, step):
        Fun.plot(self, xMin, xMax, step)
        makePlot([x for (x,y) in self.data],
                 [y for (x,y) in self.data])
```

This part of the problem is worth 5 points; we suggest that you do it only after you have finished the rest of the exam.

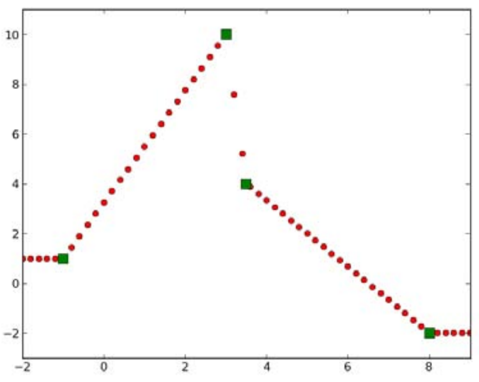
D. The `LinInterpFun` class is a subclass of `DataFun`. Its only method is `val`, which performs linear interpolation. That is, given an x_i value as input, it should look through the data to find the x_{lo} value that is, of all x values less than or equal to x_i, closest to x_i, and the x_{hi} value that is, of all x values greater than or equal to x_i, closest to x_i. Let y_{lo} be the y value associated with x_{lo} and y_{hi} be the y value associated with x_{hi}. The method should return the linear interpolation value of y for input x_i, which is:

$$y = y_{lo} + (x_i - x_{lo})\frac{(y_{hi} - y_{lo})}{(x_{hi} - x_{lo})} \quad .$$

If the query x_i is lower than any x value in the data, then the method should return the y value associated with the smallest x value. Values of x_i that are higher than any x value in the data should be handled analogously. You can assume that all numbers are floats; do not assume that the data are sorted in any way.

Here is an example plot made from an `LinInterpFun` instance. The large squares are the actual stored data points.

```
t3 = LinInterpFun([(3, 10), (-1, 1), (8, -2), (3.5, 4)])
t3.plot(-2, 10, 0.2)
```



Write your answer in the box on the next page.

Implement the `LinInterpFun` class.

```
class LinInterpFun(Fun):
    def val(self, x):
        xvals = [dx for (dx,dy) in self.data]
        xlo = min(xvals)
        xhi = max(xvals)
        if x <= xlo: return xlo
        if x >= xhi: return xhi
        for (dx, dy) in data:
            if dx < x and dx >= xlo: (xlo, ylo) = (dx, dy)
            if dx > x and dx <= xhi: (xhi, yhi) = (dx, dy)
        return ylo + (x - xlo) * (yhi - ylo)/(xhi - xlo)
```

2 State machines (10 points)

Consider the following program

```
def thing(inputList):
    output = []
    i = 0
    for x in range(3):
        y = 0
        while y < 100 and i < len(inputList):
            y = y + inputList[i]
            output.append(y)
            i = i + 1
    return output
```

A. What is the value of

```
thing([1, 2, 3, 100, 4, 9, 500, 51, -2, 57, 103, 1, 1, 1, 1, -10, 207, 3, 1])
```

```
[1, 3, 6, 106, 4, 13, 513, 51, 49, 106]
```

B. Write a single state machine class `MySM` such that `MySM().transduce(inputList)` gives the same result as `thing(inputList)`, if `inputList` is a list of numbers. Remember to include a done method, that will cause it to terminate at the same time as `thing`.

```
class MySM(sm.SM):
    startState = (0,0)
    def getNextValues(self, state, inp):
        (x, y) = state
        y += inp
        if y >= 100:
            return ((x + 1, 0), y)
        return ((x, y), y)
    def done(self, state):
        (x, y) = state
        return x >= 3
```

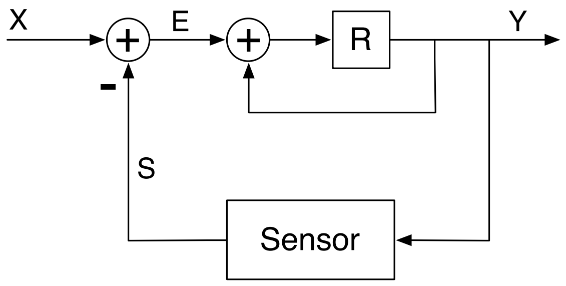
C. Recall the definition of `sm.Repeat(m, n)`: Given a terminating state machine `m`, it returns a new terminating state machine that will execute the machine `m` to completion `n` times, and then terminate.

Use `sm.Repeat` and a very simple state machine that you define to create a new state machine `MyNewSM`, such that `MyNewSM` is equivalent to an instance of `MySM`.

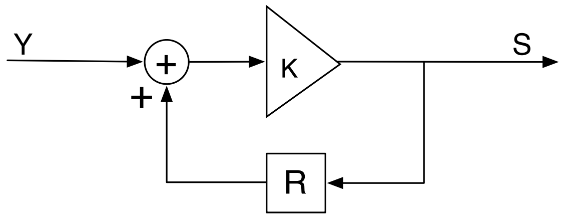
```
class Sum(sm.SM):
    startState = 0
    def getNextValues(self, state, inp):
        return (state + inp, state + inp)
    def done(self, state):
        return state > 100

myNewSM = sm.Repeat(Sum(), 3)
```

3 Linear Systems (20 points)

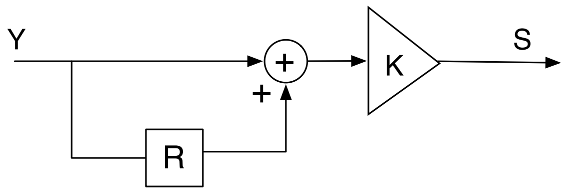
In the following system, we will consider some different possible sensors.



We can buy sensor A or sensor B as shown in the diagrams below. For each of them we can also specify the gain k at the time we order it.



Sensor A



Sensor B

A. Just considering the sensor, not yet connected into the system, if we feed a **step function** (going from 0 to 1) into a sensor A with k = 0.5, what will the output value be (approximately) after 100 steps? Assume the system starts at rest.

3/4, 7/8, 15/16, ..., 1

B. Just considering the sensor, not yet connected into the system, if we feed a **step function** (going from 0 to 1) into a sensor B with $k = 0.8$, what will the output value be (approximately) after 100 steps? Assume the system starts at rest.

1.6

C. If you plug sensor A into the overall system, what is the system function relating Y to X? (Leave k as a variable)

$$\frac{Y}{X} = \frac{R(1-kR)}{1-R+kR^2}$$

D. If we put a sensor of type B into the system, then the overall system function is

$$\frac{R}{kR^2 - R + kR + 1}$$

1. For $k = 0.25$, what are the poles of the system?

$\frac{1}{8}(3 \pm \sqrt{7}j)$

Is it stable? Yes

Does it oscillate? Yes

2. for $k = 0.1$, the poles of the system are at 0.77 and 0.13.

Is it stable? Yes

Does it oscillate? No

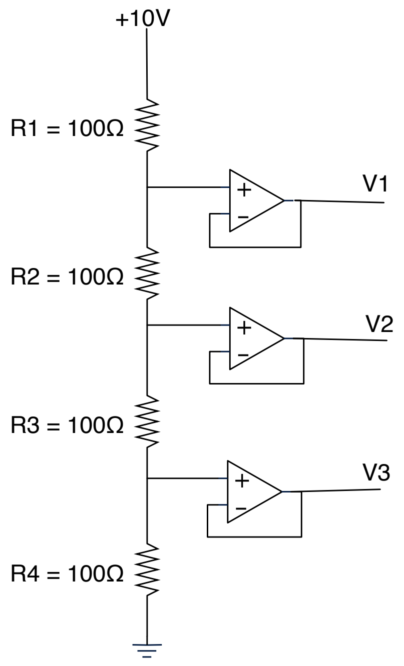
3. If you were buying a sensor of type B, which of these gains would give you faster response? Explain.

Pick $k = 0.25$ because it converges faster.

Scratch Paper

4 Circuits (20 points)

A. Dana, Emerson, and Flann have a supply of $+10$ V and want to make buffered supplies of 2.5 V, 5.0 V and 7.5 V for use in their circuit, but they are having a disagreement about how to do it. Here is the circuit Dana suggests:



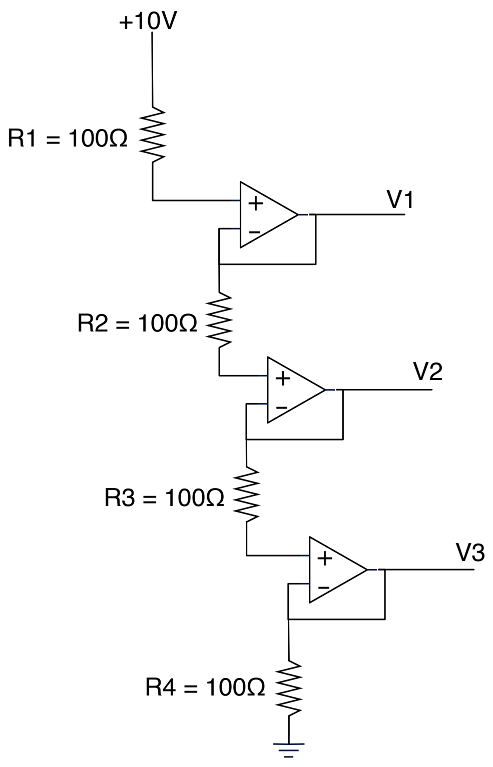
1. In Dana's circuit, what are the actual values of V_1, V_2, and V_3?

$V_1 =$ 7.5 V $V_2 =$ 5.0 V $V_3 =$ 2.5 V

If they are incorrect, can you change the resistor values to make it work? If so, how?

Correct

Here is the circuit that Emerson suggests:



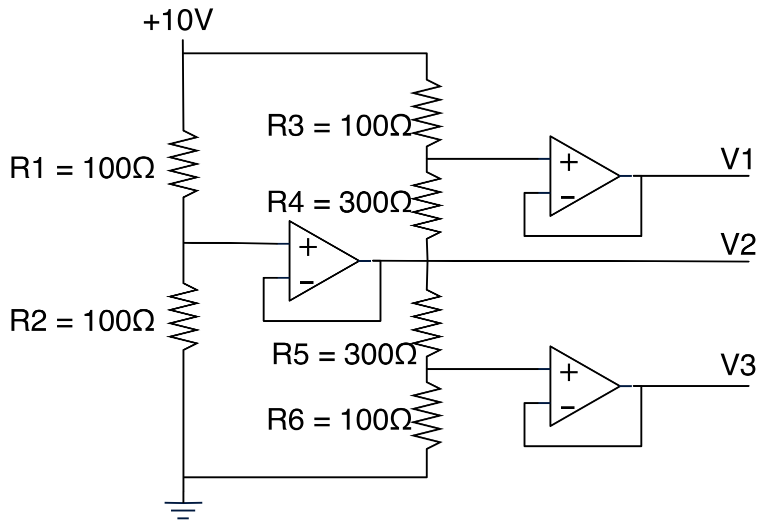
2. In Emerson's circuit, what are the actual values of V_1, V_2, and V_3?

$V_1 =$ 10 V $V_2 =$ 10 V $V_3 =$ 10 V

If they are incorrect, can you change the resistor values to make it work? If so, how?

No.

Here is the circuit that Flann suggests:



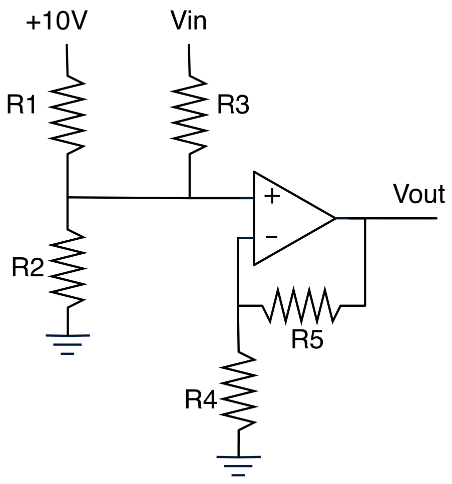
3. In Flann's circuit, what are the actual values of V_1, V_2, and V_3?

$V_1 =$ 8.75 V $V_2 =$ 5 V $V_3 =$ 1.25 V

If they are incorrect, can you change the resistor values to make it work? If so, how?

Set $R_4 = 100\,\Omega$ and $R_5 = 100\,\Omega$

B. Consider the following circuit



1. Write a formula for V_+ (the voltage on the positive input to the op-amp) in terms of V_{in} and the resistor values for this circuit:

$V_+ =$

$$\left(\frac{R_3(10R_2 + R_1V_{in})}{R_1R_3 + R_1R_2 + R_2R_3}\right)$$

2. Write a formula for V_{out} in terms of V_+ and the resistor values for this circuit:

$V_{out} =$

$$\left(\frac{R_4 + R_5}{R_4}\right) V_+$$

3. For each of these relationships, state whether it is possible to choose resistor values that make it hold in the circuit above. Write **Yes** or **No**; **it is not necessary to provide the resistor values.**

a. $V_{out} = 2.5 - \frac{3}{16} V_{in}$ No

b. $V_{out} = 2.5 + \frac{3}{16} V_{in}$ Yes

c. $V_{out} = -2.5 + \frac{3}{16} V_{in}$ No

5 State estimation (10 points)

We are interested in deciding whether the earth is in one of three states, in any given century: *warming (w), stable (s), cooling(c).*

A. We can do an experiment on an ice sample which gives one of two observations: *melting (m), not melting (n).*

Here are the observation probabilities for the experiment:

$$\Pr(O = m \mid S = w) = 0.9$$
$$\Pr(O = m \mid S = s) = 0.7$$
$$\Pr(O = m \mid S = c) = 0.1$$

Assume our initial belief state is $\Pr(S = w) = 0.4$, $\Pr(S = s) = 0.5$, $\Pr(S = c) = 0.1$.

1. What would our **belief state** be after doing the experiment and observing n?

$$\boxed{\left(\frac{1}{7}, \frac{15}{28}, \frac{9}{28}\right)}$$

B. Now, let's assume that the state of the planet is a Markov process whose transitions can be described, on the scale of centuries, as follows (of course, this is completely climatologically bogus):

		S_{t+1}		
		w	s	c
	w	0.7	0.3	0.0
S_t	s	0.4	0.2	0.4
	c	0.0	0.3	0.7

1. Circle the following sequences of states that are **possible**.

a. w, s, w, w, c, c, s, w Not possible

b. c, c, c, s, s, c, s, w Possible

c. w, s, c, w, s, c, s, w Not possible

2. If we were certain that climate was stable in some century t, and we didn't have any experimental evidence, what would our belief state about the state of the climate in century $t + 2$ be?

$t = 0$: $(0, 1, 0)$
$t = 1$: $(0.4, 0.2, 0.4)$
$t = 2$: $(0.36, 0.28, 0.36)$

6 Search (20 points)

A. We want to improve the search performance in the wolf-goat-cabbage problem (summarized below; it is the same as in the tutor problem).

- The farmer has a goat, a wolf and a head of cabbage. They come to a river (they're on the left bank) and need to get everything and everyone to the other side (the right bank).
- There's a boat there that fits at most two of them; the farmer must always be one of the two in the boat.
- If the farmer leaves the goat alone with the cabbage, the goat will eat the cabbage (so that's not a legal state). Similarly, if the farmer leaves the goat alone with the wolf... (so that's not a legal state).

Let $n(s)$ be the number of objects (wolf, goat, cabbage) that are on the incorrect side of the river in state s.

1. Andrea suggests that a good heuristic would be $n(s)-1$. Is it admissible? Why or why not?

Yes, always less than the number of steps to go.

2. Bobbie suggests that a good heuristic would be $2n(s)-1$. Is it admissible? Why or why not?

Yes, always less than or equal the number of steps to go.

3. Casey suggests that a good heuristic would be $3n(s)-1$. Is it admissible? Why or why not?

No, on the first step, $3n(s) - 1 = 9$ but there are 7 steps to go.

4. Which heuristic would be likely to reduce the search the most, while retaining optimality of the answer?

Use Bobbie's, $2n(s) - 1$, it's the largest admissible heuristic.

B. We need to travel over a network of roads through the mountains in the snow. Each road has a current condition: *clear*, *slippery*, or *buried*. There are two possible vehicles you can use: a sports car, which can only traverse clear roads or an SUV, which can traverse any road.

You start in the sports car (in location `S`), but if you are driving one vehicle, and you're in the same location as another vehicle, you can trade vehicles; if you drive your sports car to the location of the SUV (which starts in location `A`), and trade, then when you move, you will move with the SUV and the sports car will be left at that location.

We will specify the map using the data structure below, which characterizes, for each location, the roads leading out of it. Each road is described by a triple indicating the next location, the length of the road, and the condition of the road.

```
map1dist = {'S' : [('A', 2, 'clear'), ('B', 1, 'slippery')],
            'A' : [('S', 2, 'clear'), ('C', 3, 'clear'), ('D', 10, 'slippery')],
            'B' : [('S', 1, 'slippery'), ('D', 2, 'slippery')],
            'C' : [('A', 3, 'clear')],
            'D' : [('A', 10, 'slippery'), ('B', 2, 'slippery')]}
```

We are going to formulate this as a search problem with costs, to be solved using UC search. Let the cost to traverse a road just be the the length of the road times a multiplier: the multiplier is 1 for the sports car and 2 for the SUV. There is a cost of 1 for the action of swapping cars.

The possible actions are to drive on one of the roads emanating from a current location or to swap cars.

1. What information do you need to keep in each state? How will you represent it in Python?

Our location and which vehicle we are using: `(loc, car)`
We don't need to keep in the state where the other vehicle is. Once we change vehicles, we don't ever change back.

2. How would you represent the starting state (as a Python expression)?

```
('S', 'car')
```

3. What would you pass in as the second argument to `ucSearch.search`, the goal test, if the goal is to end in location `'D'`?

Write Python expression(s)

```
lambda s: s[0] == 'D'
```

4. Let the actions be described by `(action, roadNum)`, where `action` is one of `'drive'` or `'swapCars'`, and `roadNum` is an integer that means which road to drive on out of an intersection. The `roadnum` can be used as an index into the list of results in `map1dist`. When `action` is `'swapCars'`, then the `roadNum` is ignored.

If `drivingDynamics` is an instance of `sm.SM` that describes this planning domain, using your state representation, what would the output of this expression be:

```
>>> drivingDynamics.transduce([('drive', 0), ('swapCars', 0), ('drive', 1)])
```

Write a list of states.

```
{('S', 'car'), ('A', 'car'), ('A', 'suv'), ('C', 'suv')}
```

5. From that same start state, what path through state space would be found by breadth-first search, when the goal is to be in location `'D'`? Provide a list of states.

```
[('S', 'car'), ('A', 'car'), ('A', 'suv'), ('D', 'suv')]
```

6. From that same start state, what path through state space would be found by uniform-cost search? Provide a list of states.

```
[('S', 'car'), ('A', 'car'), ('A', 'suv'), ('S', 'suv'), ('B', 'suv'), ('D', 'suv')]
```

What is its cost? 13

Scratch Paper

Scratch Paper