_____级　　学号_____姓名_____

2023–2024 学年第 1 学期 Python 小测验

得分	阅卷人

I. Python basic (10 points)

For each of the expressions in the table below, write the output displayed by the interactive Python interpreter when the expression is evaluated. The output may have multiple lines (**no more than 3 lines**). If an error occurs, write "Error". The first two rows have been provided as examples.

Hints:

(a) The interactive interpreter displays the value of a successfully evaluated expression, unless it is `None`.

(b) Be careful with variables that have the same name but are bound in different environments.

(c) The `//` operator means floor division (整除).

Assume that you have started Python3 and executed the following statements:

```
def jazz (hands):
    if hands < out:
        return hands * 5
    else:
        return jazz (hands // 2) + 1

def twist (shout, it, out=7):
    while shout:
        shout, out = it(shout), print (shout, out)
    return lambda out: print (shout, out)

hands, out = 2, 3
```

Expression	Output	
`print(4, 5) + 1`	`4 5` `Error`	
`print(None, print(None))`	`None` `None None`	
`jazz(5)`	`11`	(1 point)
`(lambda out: jazz(8))(9)`	`12`	(1 point)
`twist(2, lambda x: x-2)(4)`	`2 7` `0 4`	(2 points)

_____级　　学号_____姓名_____

`twist(5, print)(out)`	`5`
	`5 7`
	`None 3`
		(3 points)

`twist(6, lambda hands: hands-out, 2)(-1)`	`6 2`
	`3 None`
	`0 -1`
		(3 points)

得分	阅卷人

II. Python OOP (10 points)

Let's build a voting tabulator for the national elections (except that we're going to assume that there are only 4 states). The definitions below initialize the instance and allow us to enter votes, by state, for each candidate. We're going to keep the votes data in the dictionary `stateVotes`. There is an entry in `stateVotes` for each state. The value stored for a state is itself a dictionary that stores the votes from that state for each candidate. A simple example, for two states (`'MA'` and `'TX'`) and two candidates, `'O'` and `'M'`, would be:

```
{'MA': {'O':100, 'M':50}, 'TX': {'O':50, 'M':100}}
```

```
class Votes:
    def __init__(self):
        self.states = ['MA', 'TX', 'CA', 'FL']
        self.candidates = ['O', 'M', 'N']
        self.stateVotes = {}
    def addVotes(self, cand, votes, state):
        if not state in self.stateVotes:
            self.stateVotes[state] = {cand : votes}
        else:
            if not cand in self.stateVotes[state]:
                self.stateVotes[state][cand] = votes
            else:
                self.stateVotes[state][cand] += votes
```

2.1 (3 points)

Write a method for the `Votes` class, called `stateWinner`, that computes the winner for a state, that is, the candidate with the most votes in that state.

Use the procedure `argmaxDict(d)`, which is called with a dictionary `d` and returns the key in `d` whose associated value is highest.

```
    def stateWinner(self, state):
        return argmaxDict(self.stateVotes[state])
```

2.2 (3 points)

Write a method for the `Votes` class, called `statesWon`, that takes a candidate as an argument and returns a list of the states that candidate won. Use the `stateWinner` method you just implemented.

```
def statesWon(self, cand):
    return [state for state in self.states if self.stateWinner(state) == cand]
```

2.3 (4 points)

Write a method for the `Votes` class, called `winnerOfMostStates`, that returns the candidate that won the most states. Use the `statesWon` method you just implemented.

For full credit, use util.argmax. If `l` is a list of items and `f` is a procedure that maps an item into a numeric score, then `util.argmax(l, f)` returns the element of `l` that has the highest score.

```
def winnerOfMostStates(self):
    return util.argmax(self.candidates, lambda cand: len(self.statesWon(cand)))
```

得分	阅卷人

III. State machines (10 points)

You are going to design a state machine that takes a string of characters as input, and outputs at each time step the largest number of times a character is repeated in a row in the sequence as observed so far. The state variable will have the following form:

`[lastCharacter, timesSeenConsecutively, best]`

that is, the state will be a list of the previous input character (except for the first time step, see below), the number of times that character has occurred consecutively up to this point, and the length of the longest sequence of repeated characters observed so far.

Complete the following definition:

```
Class SubSeq(sm.SM):
    startState = None
    def getNextValues(self, state, inp):
        if state == None:
            #part 1
        elif state[0] == inp:
            #part 2
        else:
            #part 3
```

so that it has the following example behavior:

```
>>> test = SubSeq()
>>> test.transduce('abaabcdccccc')
[1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 4, 5]
```

3.1 (3 points)

Supply the code for part 1:

```
    return ((inp, 1, 1), 1)
```

3.2 (4 points)

Supply the code for part 2:

```
    if state[1] == state[2]:
        return ((inp, state[1]+1, state[1]+1), state[1]+1)
    else:
        return ((inp, state[1]+1, state[2]), state[2])
```

3.3 (3 points)

Supply the code for part 3:

```
    return ((inp, 1, state[2]), state[2])
```