The problems below were given on a midterm several years ago. All of them concern, in one way or another, dictionaries, tuples, or multi-dimensional arrays. Try to answer them without using the computer (i.e., pretend you are taking the test). When you’ve finished with a problem, you can type the code and see the result. There is an accompanying file with fragments of code for you to try out, but don’t turn to them until you’ve tried to do the problem by hand.

1. In the Python Shell, I typed

\[
d = \{(0, 0): ['Fido', 'Rover'], (0, 1): ['Lola', 'Champ', 'Lassie'], (2, 0): ['Queenie', 'Bruiser']\}
\]

and then typed

\[
e = \text{list}(d.\text{items}())
\]

\[
\text{print}( \ e)\]

which produced the output

\[
[\((0, 0), ['Fido', 'Rover']\), \((0, 1), ['Lola', 'Champ', 'Lassie']\), \((2, 0), ['Queenie', 'Bruiser']\)]
\]

For each of the following expressions, determine both the value and the type of the expression. Possible types are dict, list, tuple, str, int. You don’t have to indicate a value for (a) and (b), as the values of these expressions are already shown above.

(a) \(d\)

(b) \(e\)

(c) \(e[0]\)

(d) \(e[0][0]\)

(e) \(e[0][0][0]\)

(f) \(d[(0,1)][1][1:3]\)

(g) \([t \text{ for } t \text{ in } d \text{ if } \text{len}(d[t]) == 2]\)
2. What are the type and value of prob2(18), where prob2 is the function defined below?

```python
def prob2(n):
    bits=[]
    while n != 0:
        bits.append(n%2)
        n=n//2
    return bits, len(bits)
```

3. The following function takes as a parameter a list of strings and returns a list of strings. (You will note that implicit in this code is a comparison of tuples. For 2-tuples (a,b) and (c,d), the rule is (a,b)<(c,d) if a<c or if a=c and b<d.)

```python
def prob3(words):
    u=[(len(word),word) for word in words]
    u.sort()
    return [w[1] for w in u]
```

(a) What is the value returned by the following call to this function?

```python
prob3(['after','alter','then'])
```

(b) Describe in general what prob3 is doing.
4. Write a function to `flatten` a two-dimensional array. That is, the function should take a list of lists of ints such as

```plaintext
[[4,1,3,-2],[6,5,4],[7,5,1,-6],[4]]
```

and return

```plaintext
[4,1,3,-2,6,5,4,7,5,1,-6,4]
```

Note that not all the inner lists have to have the same length. (There is a straightforward but slightly tedious way to do this with a doubly-nested for loop, and a very short, slick way to do it with a single for loop.)

5. A dictionary `recipe_dict` looks, in part, like this:

```plaintext
{...,'ratatouille':['eggplant','tomato','garlic',...],'
chocolate mousse':[ 'chocolate','egg whites','sugar',...],...}
```

Write an expression whose value is a list of all those dishes in the dictionary that use both eggplant and tomato. (So for example, 'ratatouille' will be in this list.) You can solve this by writing a sequence of statements whose last line is an expression that has the desired value. But this can also be done in a single line. (You can test your code by using the dictionary above; erase the ... and add another dish with eggplant and tomato. I suggest caponata.)