6.101 Recitation 11: Week 5 Recipe Midpoint **3/13/24**

This sheet is yours to keep!

Question 1: Read the problem description below, and then generate a set of all the legal phrases that could result from the "greeting" root.

All Phrases Grammar Description:

A *grammar* consists of rules that describe the syntax of legal phrases in a language (programming or otherwise).

For today, we will be using a grammar dictionary that defines the syntax rules of a tiny version of the English language. For example:

```
grammar = {
        "sentence": [["noun", "verb"], ["noun", "never", "verb"]],
        "noun": [["pigs"], ["professors"]],
        "verb": [["fly"], ["think"]],
        "greeting": [["hi", "noun"]],
        "question": [["sentence", "?"]],
    }
```

Within the grammar dictionary,

- the dictionary keys are *non-terminals* (syntax category, i.e., `"sentence"` or `"noun"`)
- any string that isn't a dictionary key is a *(terminal) word* (i.e., `"hi"` or `"fly"`)
- the dictionary values are a list of rules. Each *rule* is a list containing non-terminals and words (i.e., `["pigs",]` or `["hi", "NOUN",]`)

We want to start with a *root* string (like `"sentence"` or `"pigs"`) and expand it into a set of all possible *phrases* that can be found from the given root. We will represent a phrase using a tuple containing only terminal word strings.

Examples:

```
All phrases given the root "pigs" => {('pigs',)}
All phrases given the root  "noun" => {('pigs',), ('instructors'),}
All phrases given the root  "sentence" => {("pigs", "fly"),
("pigs", "think"),("professors", "fly"), ("professors", "think"),
("pigs", "never", "fly"), ("pigs", "never", "think"),
("professors", "never", "fly"), ("professors", "never", "think"),}

What are all the phrases that can be made given the root "greeting"?
```

Question 3: All Phrases

```
def all_phrases(grammar, root):
    """
    Using rule lists in the grammar dict, expand root into all possible
    phrases. Each phrase is a tuple of terminal word strings.
    Return a set of all valid phrases.
    """
```

```
grammar = {
       "sentence": [["noun", "verb"], ["noun", "never", "verb"]],
       "noun": [["pigs"], ["professors"]],
       "verb": [["fly"], ["think"]],
       "greeting": [["hi", "noun"]],
        "question": [["sentence", "?"]],
}
assert all_phrases(grammar, "pigs") == {("pigs",)}

expected = {("pigs", "fly", "?"), ("pigs", "think", "?"),
    ("professors", "fly", "?"), ("professors", "think", "?"),
    ("pigs", "never", "fly", "?"), ("pigs", "never", "think", "?"),
    ("professors", "never", "fly", "?"), ("professors", "never", "think", "?"),
}
assert all_phrases(grammar, "question") == expected
```

R11 Participation Credit **Kerberos : _____@mit.edu**

Hand this sheet in at the end of recitation to get participation credit for today.

Question 2:

For the `all_phrases` function described in question 3 (see page 2):

What are the base case(s)?

What are the recursive case(s)?