Tiny Practice Final

Time limit: 20 minutes

Question 1

(10 fake marks) Suppose the file `sample_imports.txt` contains a list of Python `import` statements like this:

```
import time
import random
import turtle
import doctest
import turtle
import time
```

For simplicity, assume that every line is an `import` statement formatted as in the example. There are no blank lines or extra spaces around the words.

Write a **program** that reads `sample_imports.txt` and prints the imports in *sorted order*, and with any *duplicates removed*. For example, the above file gets printed like this:

```
import doctest
import random
import time
import turtle
```

Of course, sample_imports.txt should work with any file of imports, not just the one in the example.

Marking Scheme

- **+1 mark**: opening the file
- **+1 mark**: initializing a list/dictionary
- **+1 mark**: looping over lines of the file
- **+1 mark**: remove the '\n' at the end of the read-in line
- **+1 mark**: extracting the import name from the line
- **+1 mark**: handling duplicate imports
- **+1 mark**: adding the import name to the list/dictionary
- **+1 mark**: sorting the names
- **+1 mark**: printing import statements in
- **+1 mark**: overall correct Python syntax and indentation

Up to -1 mark deducted if the program is very inefficient, or does anything unnecessary.

Notes

- If an error is very small, or maybe just a slip of the pen, you might decide to take no marks off, or only a few marks. Forgetting a single : is probably -0.5, and forgetting : multiple times is at least -1 mark. Repeated errors are more serious than one-time errors.
- Sometimes answers can be different than what the marking scheme expects. In such cases, first try to decide if it is a failing or passing answer. If it's failing, don't give more than 50%.

Sample Solutions

```
def solution1():
    """Uses a list to store modules.
    """
    module_list = []
    file = open('sample_imports.py')
    for line in file:
        line = line.strip()
        tokens = line.split(' ')
        mod = tokens[1]
        if mod not in module_list:
            module_list.append(mod)


    module_list.sort()
    for mod in module_list:
        print(f'import {mod}')
```

```python
def solution2():
    """Uses a dictionary to keep track of the modules.
    """
    module_dict = {}
    file = open('sample_imports.py')
    for line in file:
        line = line.strip()
        tokens = line.split(' ')
        mod = tokens[1]
        module_dict[mod] = 1


    all_modules = list(module_dict.keys())
    all_modules.sort()
    for mod in all_modules:
        print(f'import {mod}')
```

Question 2

(10 fake marks) Write a function called `string_sort(str_list)` that takes a list of strings as input and returns a copy of `str_list` sorted by length, from smallest to biggest. Strings of the same length should be sorted alphabetically. For example:

```
>>> string_sort(['dog', 'or', 'a', 'cat'])
['a', 'or', 'cat', 'dog']
```

Sample Solution

```
def string_sort(str_list):
    # make a list of [length, string]
    # pairs
    pairs = []
    for s in str_list:
        pairs.append([len(s), s])
    pairs.sort()

    # read the now-sorted strings
    # back into a list
    result = []
    for p in pairs:
        result.append(p[1])
    return result
```

Marking Scheme

- **+3 marks**: sorting the strings by size (do not give full marks if the code is unclear or complex)
- **+3 marks**: sorting the same-size strings alphabetically (do not give full marks if the code is unclear or complex)
- **+2 marks**: returning the strings
- **+2 mark**: overall correct Python syntax and indentation

Up to -1 mark deducted if the program is very inefficient, or does anything unnecessary.

Notes

- If an error is very small, or maybe just a slip of the pen, you might decide to take no marks off, or only a few marks. Forgetting a single : is probably -0.5, and forgetting : multiple times is at least -1 mark. Repeated errors are more serious than one-time errors.
- Sometimes answers can be different than what the marking scheme expects. In such cases, first try to decide if it is a failing or passing answer. If it's failing, don't give more than 50%.