COSC 101, Exam #3 Practice

1. For each part (a, b, and c) indicate what the program prints.

   (a) (2 points)

```
def bar(x: list) -> None:
    y = x[:]
    y.pop(1)
    y.pop(4)
    print(y)

aList = [23, 4, 'cat', 'lizards', 'adventure', False]
bar(aList)
print(aList)
```

Solution:

```
[23, 'cat', 'lizards', 'adventure']
[23, 4, 'cat', 'lizards', 'adventure', False]
```

(b) (3 points)

```python
def foo(freq: dict) -> None:
    a = freq
    print(a)
    a['hello'] += 1
    a['bye'] = 1
    print(a)

freq = {'hello': 3, 'cats': 100}
foo(freq)
print(freq)
```

Solution:

```
{'hello': 3, 'cats': 100}
{'hello': 4, 'cats': 100, 'bye': 1}
{'hello': 4, 'cats': 100, 'bye': 1}
```

(c) (3 points)

```python
def negate(words: list) -> None:
    print(words)
    for i in range(len(words)):
        words[i] = 'un'+words[i]
    print(words)

a = ["do", "apply"]
negate(a)
print(a)
```

Solution:

```
['do', 'apply']
['undo', 'unapply']
['undo', 'unapply']
```

2. For each part (a, b, and c) indicate what the program prints.

   (a) (3 points)

```
def wacky(aList: list) -> None:
    bList = aList
    print(bList)
    bList.append('tigers')
    bList.append('oh my')
    print(bList)

animals = ['cats', 'lions']
wacky(animals)
print(animals)
```

Solution:

```
['cats', 'lions']
['cats', 'lions', 'tigers', 'oh my']
['cats', 'lions', 'tigers', 'oh my']
```

(b) (3 points)

```
def plural(x: list) -> None:
    print(x)
    for w in x:
        w = w + 's'
    print(x)

a = ["pear", "apple"]
plural(a)
print(a)
```

Solution:

```
['pear', 'apple']
['pear', 'apple']
['pear', 'apple']
```

(c) (2 points)

```
def removey(x: list) -> None:
    y = x[:]
    del y[0]
    del y[3]
    print(y)

aList = ['colgate', 13, 8, 'raiders', 2, 'fig']
removey(aList)
print(aList)
```

Solution:

```
[13, 8, 'raiders', 'fig']
['colgate', 13, 8, 'raiders', 2, 'fig']
```

3. This is a two-part question. Part (a) is a helper function for Part (b). Part (b) can be completed even if you have not correctly implemented Part (a). The overall problem is about course registration.

   (a) (6 points) Write a function called `get_valid_course` that prompts the user to enter the number of a course they want to take and returns the string representing the course. The function should only return valid course numbers: course numbers with four-character department codes followed by a space and then a three digit number. For example, `'COSC 102'` is valid, but `'Computer Science 2'` is not. When a user enters an invalid course number, the function should display an error message and then reprompt them. Hint: Look at the string methods at the end of the exam.

Solution:

```
def get_valid_course() -> str:
    isValid = False
    while not isValid:
        course = input('What course do you want? ')
        if (len(course) == 8 and
            course[:4].isalpha() and
            course[4] == ' ' and
            course[5:].isdigit()):
            isValid = True
        else:
            print('You did not enter a valid course.')
    return course
```

(b) (6 points) Write a function called `register` which takes a dictionary of course enrollments as a parameter. The course enrollment dictionary has course numbers as keys and values are the number of students enrolled in the course. The user will be prompted to enter the course number for one course they wish to enroll in. (If they enter an invalid course number they will be re-prompted until they provide a valid course number.) The program will return the course the user was able to enroll in or `'FAILED'` if registration was not successful. Additionally, your function should print a message telling the user the course is not offered if the requested course is not offered (i.e., missing from the course enrollment data). Students can only enroll in a course if there is at least one seat available. **Courses allow a maximum of 24 students to enroll.** Your function should update the course enrollment dictionary to reflect the updated numbers of students in each course.

For example, if the course enrollment dictionary is originally

`{'COSC 101':22, 'COSC 102':15, 'COSC 201':24}`

and the user indicates they would like to enroll in `'COSC 101'`, the function would return `'COSC 101'` and the dictionary is now

`{'COSC 101':23, 'COSC 102':15, 'COSC 201':24}`

You are required to use the `get_course` function from part (a) and can assume the function works as described (regardless of whether your answer is correct or not).

Solution:

```
def register(enroll: dict) -> str:
    course = get_valid_course()
    if course not in enroll:
        print('Course not offered this semester')
        return 'FAILED'
    if enroll[course] > 23:
        return 'FAILED'
    enroll[course] += 1
    return course
```

4. You are given a dictionary representing the top goal scorers in the history of the FIFA World Cup. Each player is identified by their name (a string), and the corresponding number of goals they scored is represented by an integer value. Example dictionary:

```
wc_scorers = {
    "Marta": 17,
    "Miroslav Klose": 16,
    "Birgit Prinz": 14,
    "Ronaldo": 15,
    "Lionel Messi": 13
}
```

This question has two parts. The second part appears on the next page.

(a) (5 points) Write a Python function called `find_top_scorer` that takes the dictionary as input and returns the name of the player with the **most** goals in World Cup history.

Solution:

```
def find_top_scorer(wc_scorers: dict) -> str:
    top_scorer = None
    max_goals = 0
    for player, goals in wc_scorers.items():
        if goals > max_goals:
            top_scorer = player
            max_goals = goals
    return top_scorer
```

(b) (3 points) Using the function `find_top_scorer` from the previous part, write another Python function called `is_legend` that takes the dictionary of goal scorers. This function should return `True` if the top scorer has scored 15 or more goals in the World Cup, and `False` otherwise.

Example Usage 1:

```
wc_scorers_2 = {
    "Birgit Prinz": 14,
    "Lionel Messi": 13,
    "Just Fontaine": 13,
    "Kylian Mbappe": 12
}

print(is_legend(wc_scorers_2))  # Output: False
```

Example Usage 2:

```
wc_scorers = {
    "Marta": 17,
    "Miroslav Klose": 16,
    "Birgit Prinz": 14,
    "Ronaldo": 15,
    "Lionel Messi": 13
}

print(is_legend(wc_scorers))  # Output: True
```

Note: You MUST use `find_top_scorer` in `is_legend`.

Solution:

```
def is_legend(wc_scorers: dict) -> bool:
    top_scorer = find_top_scorer(wc_scorers)
    if wc_scorers[top_scorer] >= 15:
        return True
    else:
        return False
```