Mock Final Exam Answer Key

Question 1

Loop table

Given the function below:

```
def even_up_nested(lists):
  for i in range(len(lists)):
    for j in range(len(lists[i])):
      lists[i][j] += lists[i][j] % 2
  return lists
```

Complete the loop table below with the corresponding values of i, j, lists[i][j] % 2 and lists for the following functions call:

```
even_up_nested([[0, 2, 1, 3], [3, 2], [0, 1]])
```

i	j	lists[i][j] % 2	lists
0	0	0	[[0, 2, 1, 3], [3, 2], [0, 1]]
0	1	0	[[0, 2, 1, 3], [3, 2], [0, 1]]
0	2	1	[[0, 2, 2, 3], [3, 2], [0, 1]]
0	3	1	[[0, 2, 2, 4], [3, 2], [0, 1]]
1	0	1	[[0, 2, 2, 4], [4, 2], [0, 1]]
1	1	0	[[0, 2, 2, 4], [4, 2], [0, 1]]
2	0	0	[[0, 2, 2, 4], [4, 2], [0, 1]]
2	1	1	[[0, 2, 2, 4], [4, 2], [0, 2]]

Question 2

Evaluate the code below. Enter in each box what the last line of code in each chunk prints. When the code throws an error, write ERROR in the response box.

```python
# 2A
numbers = []
numbers.append(10)
numbers.append(1)
numbers.insert(1, 0)
print(numbers)
```

ANSWER 2A:
[10, 0, 1]

```python
# 2B
numbers = [1, 2]
numbers.insert(0, 3)
numbers[2] = 100
print(numbers)
```

ANSWER 2B:
[3, 1, 100]

```python
# 2C
numbers = [1, 2]
print(numbers[2])
```

ANSWER 2C:
ERROR

```python
# 2D
numbers = {1, 2, 3, 1, 2}
print(numbers)
```

ANSWER 2D:
{1, 2, 3}

```python
# 2E
numbers = {1}
numbers.add(2)
numbers.add(2)
print(numbers)
```

ANSWER 2E:
{1, 2}

Question 3

Write a program that does the following: - ask the user to a) enter the first word and b) the second word - count the number of vowels in each word - compare the counts - in main(), print a message that indicates which word has more vowels

You must decompose your program into at least two functions including the main function.

Test case 1:

```
Enter first word:
banana
Enter second word:
pear
banana has more vowels.
```

Test case 2:

```
Enter first word:
see
Enter second word:
sequoia
sequoia has more vowels.
```

Test case 3:

```
Enter first word:
cat
Enter second word:
dog
cat and dog have equal number of vowels.
```

ANSWER FOR QUESTION 3:

```
def get_vowel_count(word):
  count = 0
  for letter in word:
    if letter in "aeiou":
      count += 1
  return count

def main():
  word_1 = input("Enter first word:\n)
  word_1_count = get_vowel_count(word_1)
  word_2 = input("Enter second word:\n)
  word_2_count = get_vowel_count(word_2)

  if word_1_count > word_2_count:
    print(word_1 + " has more vowels.")
  elif word_2_count > word_1_count:
    print(word_2 + " has more vowels.")
  else:
    print(word_1 + " and " + word_2 +
    " have equal number of vowels.")
```

Question 4

Write a Python function called trim_ends that has a 2D list as a parameter. The function should mutate and return the argument list, removing the first and last element of each sublist (if the sublist is not empty).

Test case:

```
numbers = [ [10, 20, 200, 40],
            [ ], [10],
            [1000, 1000, 10],
            [20, 30, 4, 100] ]
trim_ends(numbers)
assert numbers == [[20, 200], [ ], [ ], [1000], [30, 4]]
```

ANSWER FOR QUESTION 4:

```
def trim_ends(lists):
  for sublist in lists:
    if len(sublist) > 0:
      sublist.pop(0)
    if len(sublist) > 0:
      sublist.pop(-1)
```

Question 5

Write a python function that does the following:

1. Its name is create_list
2. It takes two arguments, a set of strings and an integer n
3. It returns a list that contains each string from the set repeated n times

```
items = {"banana", "apple", "pear"}
assert create_list(items, 2) == ['banana', 'banana',
                        'apple', 'apple', 'pear', 'pear']
```

ANSWER FOR QUESTION 5:

```
# solution 1
def create_list(items, n):
    new_list = [ ]
    for value in items:
      for i in range(n):
        new_list.append(value)
    return new_list
```

```
# solution 2
def create_list(items, n):
    new_list = [ ]
    for value in items:
      new_list.extend([value] * n)
    return new_list
```

Question 6

The python code and the contents of the file named data.txt. The python code writes content to a file named result.txt. You must determine what the contents of result.txt will be after the code runs. Put your answer in the response box.

data.txt

```
one silver edging
trees leaves are green
this simple request is finally
a moody final countdown
```

```
def is_acceptable(x):
  for i in range(0, len(x)-1):
    if x[i] == x[i+1] and x[i] in "aeiou":
      return True
  return False

def main():
  data = open('data.txt', 'r')
  result = open('result.txt', 'w')
  for line in data:
    words = line.strip('\n').split(' ')
    for word in words:
      z = is_acceptable(word)
      if z:
          result.write(word + '\n')
  data.close()
  result.close()
main()
```

ANSWER FOR QUESTION 6:
trees
green
moody

Question 7

Write a function called star_consonants that has one string as parameter. The function returns a new string of the same length as the parameter string, with every consonant replaced by an asterisk ("*").

```
assert star_consonants("banana") == "*a*a*a"
assert star_consonants("a") == "*a"
assert star_consonants("apple") == "a***e"
assert star_consonants("") == ""
```

ANSWER FOR QUESTION 7:

```
def star_consonants(string):
  new_string = ""
  for char in string:
    if char.lower() not in "aeiou":
      new_string += "*"
    else:
      new_string += char
  return new_string
```

Question 8

Write a function called total that has one parameter named file_name, being the name of a file to read. The function expects that the file to read has one or more integer numbers on it per line. It iterates over the lines and numbers to compute the total of all the numbers from the file. It returns the total.

Example of data.txt file

```
5 10 2
1 0
5 1
20
```

```
assert total("data.txt") == 44
```

ANSWER FOR QUESTION 8:

```
def total(file_name):
  total = 0
  f = open(file_name, "r")
  for line in f:
    numbers = line.strip().split()
    for n in numbers:
      total += int(n)
  return total
```

Question 9

Write a function that does the following:

1. Its name is average_rows
2. It has one parameter named lists, being a 2D list of float numbers
3. For each list (row) within the 2D list, it should calculate the average of the numbers within, round it at two decimals, and place the resulting average in a new list at the same index
4. It returns the list of the averages

```
assert average_rows([[1.2, 5.4, 4.3, 2.0],
                     [0.0. 1.0]]) == [3.23, 0.5]
assert average_rows([[], [10.5]]) == [None, 10.5]
assert average_rows([[1.0], [2.5, 3.5, 4.5], [0.0, 0.0],
                     [0.0, 2.0]]) == [1, 3.5, 0.0, 1.0]
```

ANSWER FOR QUESTION 9:

```
def average_rows(lists):
  averages= [ ]
  for row in lists:
    if len(row) > 0:
      total= 0
      for number in row:
        total += number
      this_row_average= round(total/len(row), 2)
      averages.append(this_row_average)
    else:
      averages.append(None)
return averages
```

Question 10

Write a function called mutate_dict that takes two arguments: a dictionary with string keys and integer values, and a set of strings. The function mutates and returns the dictionary argument adding the strings in the set as keys in the dictionary:

1. If the key already exists in the dictionary, do not change anything
2. If the key does not exist in the dictionary, create with the value zero associated with it

```
test_dictionary = {"z": 1, "x": 2, "r": 20}
mutate_dict(test_dictionary, {"a", "z", "r", "b"})
assert test_dictionary == {"z": 1, "x": 2,
                           "r": 20, "a": 0, "b": 0}
```

ANSWER FOR QUESTION 10:

```
def mutate_dict(dictionary, string_set):
  for key in string_set:
    if key not in dictionary:
      dictionary[key] = 0
  return dictionary
```

Question 11

Write a Python function called remove_vowel_ending that takes a list of strings as argument (you can assume strings are never empty). The function should remove list items that end in a vowel (check for upper or lower case).

```
test_list = ["Peter", "Bob", "Ana", "MARIO", "CEDRIC"]
remove_vowel_ending(test_list)
assert test_list == ["Peter", "Bob", "CEDRIC"]

assert remove_vowel_ending([]) == []
```

ANSWER FOR QUESTION 11:

```
def remove_vowel_ending(strings):
  for i in range(len(strings)-1, -1, -1):
    if strings[i][-1].lower() in "aeiou":
      strings.pop(i)
  return strings
```

Question 12

Write a Python function called remove_vowels that takes a list of strings as arguments. The function should mutate and return the argument list, removing the vowels of each item in the list.

```
test_list = ["Peter", "Bob", "Ana", "MARIO", "CEDRIC"]
remove_vowels(test_list)
assert test_list == ["Ptr", "Bb", "n", "MR", "CDRC"]

assert remove_vowels([]) == []
```

ANSWER FOR QUESTION 12:

```
def remove_vowels(strings):
  for i in range(len(strings)):
    new_string = ""
    for char in strings[i]:
      if char.lower() not in "aeiou":
        new_string += char
    strings[i] = new_string
  return strings
```

Question 13

Write a python function that takes a list of integers representing years, and evaluates whether each year (for example, 2024) is a leap year or a regular year. The function should return a dictionary with the results.

Leap Years are:

- Divisible by 4 and not divisible by 100
- Divisible by 100 and also divisible by 400

All other cases are regular years. Test cases (your leap_year function definition should work with these calls):

```
years [1992, 2000, 1900, 1700, 2024]
result = leap_year(years)
assert result == {1992: 'Leap Year',
                  2000: 'Leap Year',
                  1900: 'Regular Year',
                  1700: 'Regular Year',
                  2024: 'Leap Year'}
```

ANSWER FOR QUESTION 13:

```
def is_leap_year(y):
  if y % 4 == 0 and y % 100 != 0:
    return "Leap Year"
  elif y % 400 == 0:
    return "Leap Year"
  return "Regular Year"

def leap_year(years):
  result = {}
  for y in years:
    result[y] = is_leap_year(y)
  return result
```

Question 14

Write python code that given a list of years, it mutates the list by removing the leap years. All your code should be in functions.

ANSWER FOR QUESTION 14:

```
def is_leap_year(y):
  if y % 4 == 0 and y % 100 != 0:
    return "Leap Year"
  elif y % 400 == 0:
    return "Leap Year"
  return "Regular Year"

def remove_leap_year(years):
  for i in range(len(years)-1, -1, -1):
    if is_leap_year(years[i]):
      years.pop(i)
  return years
```