Selected Practice Questions Answers

Understanding Code

```
[[1, 2, 3], 6, 7]
```

This question tests if the student understands the syntax for `if-else` clauses and also list manipulation.

-2 marks for incorrectly nested lists (i.e. `[1, 2, 3, 6, 7]`.

A

B

```
(2, 4, 1, 2, 3, 4)
```

This question tests if the student understands truth values and tuple slicing.

-2 marks for wrong slicing.
-2 marks for wrong if-else branch.

C

```
13.5
```

This question tests if the student understands the `for` loop and the `%`, and `/` operators.

-1 mark for returning 13.
-2 marks for returning 8.5 (skipping i=5) with reasonable workout.

D

```
21
```

This question tests if the student understands the passing of arguments in functions and variable scoping.

-2 marks for returning `16` with reasonable workings (where `y` is incorrectly identified as 5 instead of 10). Note that, in the return statement of `bar()`, `x` must be 1 and `z` must be 10.

E

```
Bad things!
Finished!
```

This question tests if the student understands Exception handling syntax.

-2 marks for any incorrect responses printed out.

F

```
[1, 2, 3]
```

This question tests if the student understands aliasing.

Recursion & Iteration

A. **[Warm Up]** Write an **recursive** function `s11(n)` that returns the value for $s_{11}(n)$. [4 marks]

```
def s11(n):
    if n == 1:
        return 1
    elif n%2 == 0:
        return s11(n-1) - n
    else:
        return s11(n-1) + n
```

In this question and thereafter:
- full mark for a completely correct answer
- partial mark for a partially correct answer (e.g. -1 mark for incorrect base case, -1 for incorrect indentation, -1 for syntax errors)
- zero mark for a completely incorrect answer or no answer

C. Write an **iterative** function `s11(n)` that returns the value for $s_{11}(n)$. [4 marks]

```
def s11(n):
    total = 0
    for i in range(1,n+1):
        if i%2 == 0:
            total -= i
        else:
            total += i
    return total
```

E. Write a function `s21(n)` that returns the value for $s_{21}(n)$. [5 marks]

```
def s21(n):
    total = 0
    for i in range(1,n+1):
        if (i-1)%3 < 2:
            total += i
        else:
            total -= i
    return total
```

F. Write a function `make_s(i,j)` that returns the function $s_{ij}(n)$. In other words, we could have defined `s11(n)` for Part (A) (or (C) depending on your implementation) as follows:

```
s11 = make_s(1,1)
s12 = make_s(1,2)
```

[5 marks]

```
def make_s(i,j):
    def helper(n):
        total = 0
        for e in range(1,n+1):
            if (e-1)%(i+j) < i:
                total += e
            else:
                total -= e
        return total
    return helper
```

List Manipulation

Q1

```
def transpose_matrix(list_of_lists):
    transposed_list = []
    for i in range(len(list_of_lists[0])): #note its the other way round
                                           # since you want to swap rows with cols
        column = []
        for j in range(len(list_of_lists)):
            curr_element = list_of_lists[j][i]
            column.append(curr_element)
        transposed_list.append(column)
    return transposed_list

def transpose_matrix_one_line(list_of_lists):
    return  [[row[i] for row in list_of_lists] for i in range(len(list_of_lists[0]))]
```

Q2

```
def match(item, seq):

    # Filter
    result = list(filter(lambda x: x if x == item else None, seq))

    # Count
    count = len(result)
    return (result, count)
```

List & Dictionary Manipulation

Q1

```
def count_sentence(sentence):
    letter_count = len(sentence)-1
    for w in sentence:
        letter_count += len(w)
    return [len(sentence), letter_count]

cs1010s = [['C', 'S', '1', '0', '1', '0', 'S'], ['R', 'o', 'c', 'k', 's']]
python = [['P', 'y', 't', 'h', 'o', 'n'], ['i', 's'], ['c', 'o', 'o', 'l']]
##print(count_sentence(cs1010s))
##print(count_sentence(python))

def letter_count(sentence):
    letters = []
    letters_count = []
    while sentence:
        word = sentence.pop(0)
        while word:
            char = word.pop(0)
            if char in letters:
                letters_count[letters.index(char)] += 1
            else:
                letters.append(char)
                letters_count.append(1)

    ## merge the two lists
    new_lst = []
    for i in range(len(letters)):
        current_letter = letters[i]
        current_letter_count = letters_count[i]
        new_lst.append([current_letter,current_letter_count])
    return new_lst

cs1010s = [['C', 'S', '1', '0', '1', '0', 'S'], ['R', 'o', 'c', 'k', 's']]
python = [['P', 'y', 't', 'h', 'o', 'n'], ['i', 's'], ['c', 'o', 'o', 'l']]
##print(letter_count(cs1010s))
##print(letter_count(python))
```

```
def most_frequent_letters(sentence):
    letter_count_num = letter_count(sentence)

    # find max value of 2nd element of list
    max_val = find_max(letter_count_num,1)

    most_freq_letters = []
    for entry in letter_count_num:
        if entry[1] == max_val:
            most_freq_letters.append(entry[0])

    return most_freq_letters


def find_max(lst,col_index):
    max_val = 0
    for i in range(len(lst)):
        current_element = lst[i][col_index]
        if current_element > max_val:
            max_val = current_element
    return max_val
```

Q2

```
def sieve(item, seq):

    # Create dictionary
    new_dict = dict(map(lambda x: x, seq))

    # Filter
    if item in new_dict:
        del new_dict[item]

    return new_dict
```

Logic

Q1

```
message = "TOILET"

def puzzle(message):
    count_p = sum([1 if message[i] != 'T' else 0 for i in range(0, len(message), 3)])
    count_e = sum([1 if message[i] != 'O' else 0 for i in range(1, len(message), 3)])
    count_r = sum([1 if message[i] != 'M' else 0 for i in range(2, len(message), 3)])
    total = count_p + count_e + count_r
    return total
```

```python
def puzzle_one_liner(message):
    return sum([1 if (message[i] != z) else 0 for k,z in list(enumerate(['T','O','M'])) for i in range(k,len(message),3)])
```

Q2

```python
def find_combined_median(l1,l2):
    l3 = l1 + l2
    l3.sort()
    middle_num_index = len(l3) // 2

    if l3 == []: #if empty list, do nothing.
        return

    if len(l3) == 1: #if 1, return the first element
        return l3[0]

    if len(l3) % 2 != 0: #if odd numbered
        return l3[middle_num_index]

    median = (l3[middle_num_index] + l3[middle_num_index - 1])/2
    return median
```