THE UNIVERSITY OF AUCKLAND

SEMESTER TWO, 2017
Campus: City

COMPUTER SCIENCE

Principles of Programming

(Time Allowed: TWO hours)

SOLUTIONS

NOTE:
You must answer **all** questions in this exam.
No calculators are permitted.
Answer in the space provided in this booklet.
There is space at the back for answers which overflow the allotted space.

Surname	
Forenames	
Preferred Name (if different to forenames)	
Student ID	
Username	

Q1 (/12)	**Q4** (/15)	**Q7** (/15)
Q2 (/14)	**Q5** (/9)	**Q8** (/10)
Q3 (/15)	**Q6** (/10)	**TOTAL** (/100)

Question 1 (12 marks)

a) Complete the output produced by the following code.

```
a = 21 / 2 // 3.0 + 3 ** 2 // 4
b = 4 % 5 + 4 % 4
print("a:", a, "b:", b)
```

```
a:   5.0   b:   4
```

(2 marks)

b) In the code below, assign two integer values to the variable, `value`, so that the output when the code is executed is:

```
Yes
No
```

```
value = 15
if len(str(value)) == 2 and value % 3 == 0 or value % 5 == 0:
    print("Yes")
else:
    print("No")
```

```
value = 101
if len(str(value)) == 2 and value % 3 == 0 or value % 5 == 0:
    print("Yes")
else:
    print("No")
```

(3 marks)

c) Complete the output produced by the following code.

```
words = "FROWNING IS SMILING UPSIDE DOWN"
index1 = words.find("ING")
index2 = words.rfind("ING")
a_word = words[index1: index2: 4]
print("Word:", a_word)
```

```
Word:  IIM
```

(2 marks)

d) Complete the code below which prompts the user for the answer to the problem: " 8 * 3 = ". The completed code executes as shown in the following two example outputs (the user input is shown in a larger bold font):

```
Give the answer
  8 * 3 = 24
Correct
```

```
Give the answer
  8 * 3 = 5
Incorrect
```

```
answer = 24
comment = "Give the answer"
prompt = "  8 * 3 = "
```

```
print(comment)
user_num = int(input(prompt))
```

(2 marks)

```
if user_num == answer:
   print("Correct")
else:
   print("Incorrect")
```

e) Given the `some_ifs()` function below, give the output produced by the function call:

```
some_ifs(23, 41)
```

```
def some_ifs(num1, num2):
   if num1 > 20 and num1 < 41:
      print("A", end= " ")

      if not (num2 < 20 or num1 > num2):
          print("B", end= " ")
      elif num2 < 50:
          print("C", end= " ")

      if num1 > 20 and num1 > num2:
          print("D", end= " ")
      else:
          print("E", end= " ")

      print("F", end= " ")
   else:
      print("G", end= " ")2

   print("H", end= "")
```

```
A B E F H
```

(3 marks)

Question 2 (14 marks)

a) In the `main()` function below, complete the function call so that the output when the `main()` function is executed is:

```
sat catcatcat the
```

```
def main():
    print_result("the cat sat", 3)
```

(3 marks)

```
def print_result(phrase, how_many):
    words = phrase.split()
    the_word = words[1]
    the_word = the_word * how_many
    print(words[2], the_word, words[0])
```

b) Complete the `get_funny_sum()` function below which has two integer parameters. The function returns the sum of the leftmost digit of each of the two numbers. For example, executing the following code with the completed function:

```
print("1.", get_funny_sum(123435, 21876))
print("2.", get_funny_sum(3135, 10876789))
print("3.", get_funny_sum(13, 12))
print("4.", get_funny_sum(135, 523677))
```

prints:

```
1. 3
2. 4
3. 2
4. 6
```

```
def get_funny_sum(num1, num2):
    left_two1 = str(num1)[:1]
    left_two2 = str(num2)[:1]
    total = int(left_two1) + int(left_two2)
    return total
```

(5 marks)

c) Complete the `get_new_word()` function which has two string parameters. If the second parameter string is the same as some part of the first parameter string, the function returns the string which results from removing the second parameter string from the first parameter string. If the second parameter string is not the same as some part of the first parameter string, the function returns the first parameter string unchanged. You can assume that there is at most one occurrence of the second string in the first string. For example, executing the following function calls with the completed function:

```
print("1.", get_new_word("happiness", "pin"))
print("2.", get_new_word("wordiness", "wordiness"))
print("3.", get_new_word("holidays", "days"))
print("4.", get_new_word("love", "glove"))
```

prints:

```
1. hapess
2.
3. holi
4. love
```

```
def get_new_word(word1, word2):
```

```
    new_word = word1

    if word2 in word1:
        index = word1.find(word2)
        new_word = word1[:index] +
                   word1[index + len(word2):]

    return new_word
```

(6 marks)

Question 3 (15 marks)

a) Complete the following `for ... in range(...)` loop so that the output when the code is executed is:

```
4 9 14 19 24
```

```
for num  in range( 4, 25, 5 ):
     print(num, end = " ")
```

(2 marks)

b) Complete the following `for ... in range(...)` loop so that the output when the code is executed is:

```
10 7 4 1 -2 -5
```

```
for value   in range( 10, -6, -3):
     print(value, end = " ")
```

(2 marks)

c) Give the output produced when the following code is executed.

```
count1 = 9
count2 = 0
count3 = 0

while count1 > 2:
   count2 += 1
   if count1 % 4 == 0:
      count3 += 1
      count1 = count1 - 2
   count1 = count1 - 1
   print(count1, count2, count3)
```

```
8 1 0
5 2 1
4 3 1
1 4 2
```

(3 marks)

d) Complete the output produced when the following `main()` function is executed.

```
def main():
    print("Score:", get_score('3247643'))

def get_score(nums_str):
    score = 0
    for index in range(2, len(nums_str), 3):
        digit1 = int(nums_str[index - 2])
        digit2 = int(nums_str[index - 1])
        if digit1 > digit2:
            score = score + 5
    return score
```

```
Score: 10
```

(3 marks)

e) Complete the `get_string_of_random_digits()` function below which returns a string made up of the random digits 1, 2, 3, 4, 5 and 6. The function stops generating random digits as soon as the first 6 is generated, i.e., the last digit of the string returned by the function is always a 6. Assume that the `random` module has been imported. For example, executing the following function calls with the completed function:

```
print("1.", get_string_of_random_digits())
print("2.", get_string_of_random_digits())
print("3.", get_string_of_random_digits())
print("4.", get_string_of_random_digits())
```

may print:

```
1. 256
2. 223226
3. 26
4. 45356
```

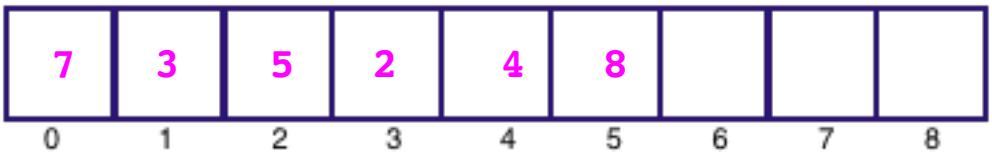
```
def get_string_of_random_digits():
    random_digits = ""
    random_num = random.randrange(1, 7)
    while random_num != 6:
        random_digits = random_digits +
                                str(random_num)
        random_num = random.randrange(1,7)
    return random_digits + "6"
```

(5 marks)

Question 4 (15 marks)

a) In the boxes below, show each element of `a_list` after the following code has been executed. Use as many of the boxes as you need.

```
a_list = [7, 6, 3, 4, 2]

value = a_list.pop()
a_list.insert(3, value)
a_list.pop(1)
a_list.insert(2, 5)
value = 2 * a_list.index(4)
a_list.append(value)
```



(4 marks)

b) Complete the output produced when the following `main()` function is executed.

```
def main():
    total = 0
    letter1 = "A"
    letter2 = "S"
    letter3 = "Z"

    a_list = ["", "XYZ - 3", "TUV - 4", "ABC - 1", "GHI - 4",
              "QRS - 2", "NOP - 5"]

    for index in range(1, len(a_list)):
        value = a_list[index]
        if letter1 in value or letter2 in value or letter3 in value:
            print(index, total)
            total =  total + int(value[-1]) * index

    print("Final total:", total)
```

```
1 0
3 3
5 6
Final total: 16
```

(5 marks)

c) Complete the `remove_1s_and_3s()` function below which is passed a list of integers as a parameter. The function removes any elements which have the value 1 or 3 from the parameter list. For example, executing the following code with the completed function:

```
digits = [1, 2, 5, 4, 3, 4]
remove_1s_and_3s(digits)
print("1.", digits)

digits = [5, 2, 9]
remove_1s_and_3s(digits)
print("2.", digits)

digits = [2, 5, 4, 3, 4]
remove_1s_and_3s(digits)
print("3.", digits)
```

prints:

```
1. [2, 5, 4, 4]
2. [5, 2, 9]
3. [2, 5, 4, 4]
```

```
def remove_1s_and_3s(nums):
```

```
for index in range(len(a_string)-1,-1,-1):
    digit = a_string[index]
    if digit == 1 or digit == 3:
        a_string.pop(index)
```

(6 marks)

Question 5 (9 marks)

a) Complete the output produced when the following `main()` function is executed.

```
def main():
    tuple1 = (2, 4, 1)
    fiddle1(tuple1)
    print("Tuple:", tuple1)

def fiddle1(a_tuple):
    a_list = list(a_tuple)
    a_list.append(4)
    a_list.append(5)
    a_tuple = tuple(a_list)
```

```
Tuple: (2, 4, 1)
```

(2 marks)

b) Complete the output produced when the following `main()` function is executed.

```
def main():
    a_list = [3, 2, 5]
    fiddle2(a_list)
    print("List:", a_list)

def fiddle2(list1):
    list2 = []
    list2.append(list1[0])
    list2.append(list1[1])
    list1[1] = list2[-2] + 1
    list1 = list2
```

```
List: [3, 4, 5]
```

(2 marks)

c) Given the following code, what is the type of each of the three Python objects (`object1`, `object2` and `object3`)?

```
a_list = ['a', 'one', 5, 4]
a_tuple = (a_list[2], a_list[0], "five")
a_dict = {'a': [3, 2], 3: [3]}

object1 = a_tuple[1] * a_list[2]
object2 = a_dict[3].pop(0)
object3 = [a_list[0]] + a_dict['a']
```

```
object1 is of type: str
object2 is of type: int
object3 is of type: list
```

(3 marks)

d) The following function contains a docstring. In the docstring, add **one** doctest which does not fail.

```
def get_number(values):
    """ Processes a list of numbers
```

```
>>> get_number([3, 4, 1, 5, 6])
4
```

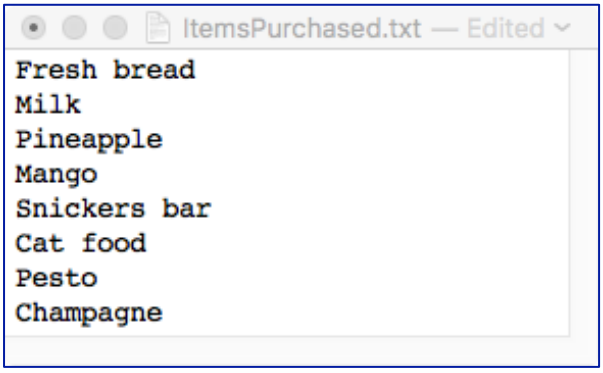
(2 marks)

```
    """

    number = len(values) // 2
    values.sort()

    return values[number]

import doctest
doctest.testmod()
```

Question 6 (10 marks)

The following program reads information from the input file, `"ItemsPurchased.txt"` (a list of grocery item names) and writes a docket to the output file, `"GroceryDocket.txt"`. The docket writes out each item, the item price and the total price.

Below is an example of an `"ItemsPurchased.txt"` file (on the left) and the corresponding `"GroceryDocket.txt"` file (on the right) produced by the completed program:



a) Complete the `get_list_of_items()` function which is passed one parameter, the name of a file which contains the names of all the items purchased. This function returns a list of all the lines of text from the file. Each element of the returned list is a string and it should not contain any newline characters.

b) Complete the `write_docket()` function which has two parameters: the name of the output file and the list of items purchased (strings). This function writes a numbered list (starting from the number 1) of each item from the parameter list. Each line of the output file contains the item name obtained from the parameter list, followed by `": $"`, followed by the price. In order to get the price for each item your code **MUST** make a call to the `get_item_price()` function which returns the price corresponding to the item name. As well, the final four lines of the output file are:

- a blank line,
- a line of three spaces followed by 15 `"="` symbols,
- a line with the string `"  Total cost: $"` followed by the total of all the items on the list rounded to the closest whole number,

and,

- a line of three spaces followed by 15 `"="` symbols.

See the screenshot of the example output file above on the right.

Note: you can assume that the `get_item_price()` function will always return a float number.

```
def main():
    items_list = get_list_of_items("ItemsPurchased.txt")
    write_docket("GroceryDocket.txt", items_list)
```

```
def get_list_of_items(filename):
```

```
file_in = open(filename, 'r')
contents = file_in.read()
file_in.close()
list_of_items = contents.split("\n")

return list_of_items
```

```
def get_item_price(item_name):
```

```
item_prices_dict =    {'Fresh bread': 3.99,
                'Milk': 4.8,
                'Pineapple': 3.6,
                'Mango': 1.89,
                'Snickers bar': 1.8,
                'Cat food': 2.75,
                'Pesto': 9.44,
                'Champagne': 15.65,
                }
return item_prices_dict[item_name]
```

```
def write_docket(filename, items):
```

```
price_str = "   Total cost: $"

file_out = open(filename, 'w')
total = 0
number = 1
for item_name in items:
    price = get_item_price(item_name)
    total = total + price
    file_out.write(str(number) + ". " +
                           item_name + ": $" +
                            str(price) + "\n")
    number = number + 1

file_out.write("\n    " + "=" * 15)
file_out.write("\n   Total cost: $" +
                      str(round(total)))
file_out.write("\n    " + "=" * 15)
file_out.close()
```

```
main()
```

(10 marks)

Question 7 (15 marks)

a) Complete the output produced when the following `main()` function is executed:

```
def main():
   a_dict = {3:5, 2:6, 12:9, 5:2}
   for key in a_dict:
      a_dict[key] = a_dict[key] + key

   print("Dictionary:", a_dict)
```

```
Dictionary: {3: 8, 2: 8, 12: 21, 5: 7}
```

(4 marks)

b) Give the output produced when the following `main()` function is executed:

```
def main():
   a_dict = {115 : ['Xymenes', 'Fran', 'Joe', 'Fazeel'],
          116 : ['Cynthia', 'Moh', 'Ivy', 'Etti', 'Abel'],
          132 : ['Phoebe', 'Reagan', 'Sue', 'Kate', 'Dana'],
          205 : ['Sid', 'Ursula', 'Gen', 'Ben'],
          356 : ['Jill', 'Merril', 'Marlo'],
          221 : ['Kathi', 'Sue', 'Val']
   }

   person = "Merril"
   the_key = -1
   for key in a_dict:
      if person in a_dict[key]:
          index = a_dict[key].index(person)
          a_dict[key].pop(index)
          the_key = key

   if the_key == -1:
      print("No result")
   else:
      print(a_dict[the_key])
```

```
['Jill', 'Marlo']
```

(4 marks)

c) Complete the `get_dict()` function which is passed a list of words as a parameter. The function creates and returns a dictionary object. The keys of the dictionary are the number of vowels in a word and the corresponding values are a list of all the UNIQUE words which contain that number of vowels. Your code **MUST** make a call to the `get_num_vowels()` function which is passed a string and returns the number of vowels in the string. For example, executing the following `main()` function with the completed function prints:

```
1 : ['word', 'world', 'happy', 'sixty']
2 : ['seven', 'heart', 'pizza']
3 : ['about', 'again']
```

```
def main():
   word_list = ['word', 'seven', 'world', 'about', 'heart',
      'heart', 'pizza', 'happy', 'again', 'sixty', 'word']
   num_vowels_dict = get_dict(word_list)

   for key in num_vowels_dict:
      print(key, ":", num_vowels_dict[key])
```

```
def get_dict(word_list):
```

```
a_dict = {}

for each_word in word_list:
 num_vowels = get_num_vowels(each_word)

 if num_vowels in a_dict:
      if each_word not in a_dict[num_vowels]:
          a_dict[num_vowels].append(each_word)
 else:
   a_dict[num_vowels] = [each_word]

return a_dict
```

(7 marks)

```
def get_num_vowels(word):
   vowels = "aeiouAEIOU"
   count = 0
   for letter in word:
      if letter in vowels:
          count = count + 1

   return count
main()
```

Question 8 (10 marks)

Parts a) and b) of this question refer to the following program:

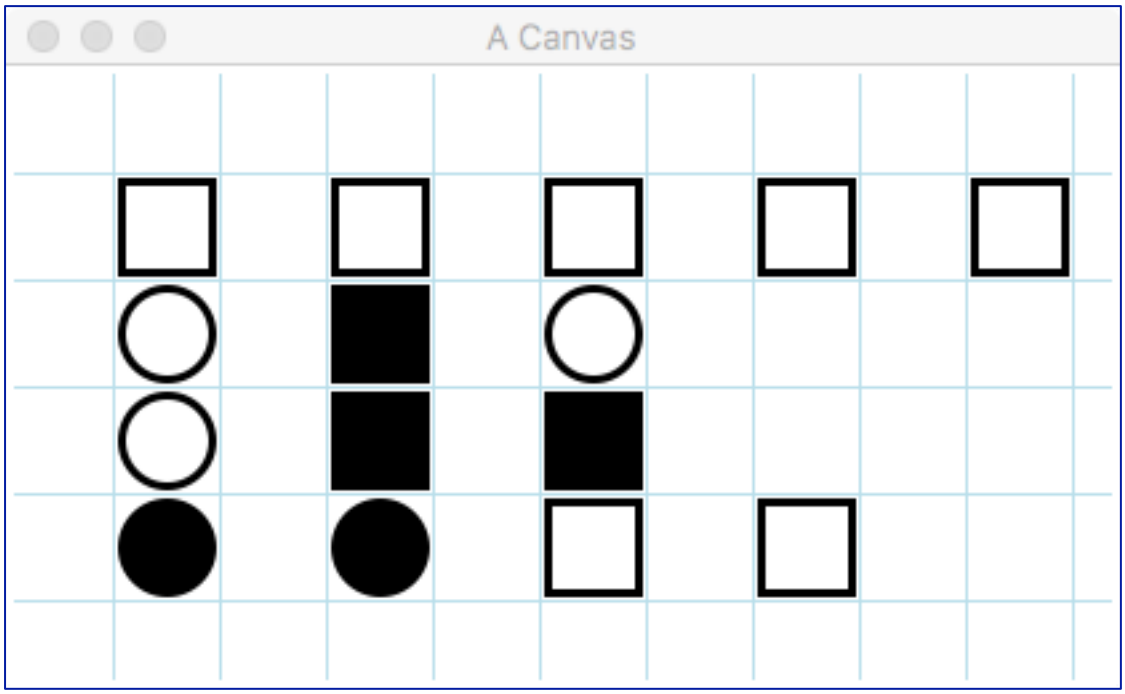
```
from tkinter import *

def draw_pattern(a_canvas):
    size = 10
    numbers = [5, 3, 3, 4]
    shapes = 'aaaaabcbbccddaa'
    shape_index = 0
    top = size

    for row in range(len(numbers)):
        num_per_row = numbers[row]
        left = size
        for col in range(num_per_row):
            symbol = shapes[shape_index]
            shape_index = shape_index + 1
            area = (left, top, left + size, top + size)
            if symbol == "a":
                a_canvas.create_rectangle(area)
            elif symbol == "b":
                a_canvas.create_oval(area)
            elif symbol == "c":
                a_canvas.create_rectangle(area, fill='black')
            elif symbol == "d":
                a_canvas.create_oval(area, fill='black')
            left =  left + size * 2

        top = top + size

def main():
    root = Tk()
    root.title("A Canvas")
    root.geometry("255x155+10+10")
    a_canvas = Canvas(root, bg="white")
    a_canvas.pack(fill=BOTH, expand=1) #Canvas fills whole window

    draw_pattern(a_canvas)
    root.mainloop()

main()
```

a) In the above program, how many rows of shapes are drawn?

Four rows

(2 marks)

b) As accurately as possible, in the window below, show what is drawn by the above program. Grid lines have been drawn in the window to help you. The gap between adjacent gridlines is 10 pixels.



(8 marks)