

Question 1 (12 marks)

- a) Complete the output produced by the following code.

```
a = 5 // 3 + 6 % 8
b = 5.7 // 3 + 16 % 4
print("a:", a, "b:", b)
```

a: b:

(2 marks)

- b) Complete the output produced by the following code.

```
result = 21 // 4 ** 2 - 1 + 16 / 5 * 3 // 2 - 1
print("Result:", result)
```

Result:

(2 marks)

- c) Complete the output produced by the following code.

```
phrase = "How marvellous."
bits_of_string = (phrase[-2] + "-" + phrase[2: 5] +
                 "-" + phrase[4: : 2])
print("Output:", bits_of_string)
```

Output:

(2 marks)

- d) Assume the variable,
- `value`
- , has been initialised to an integer value. Write a boolean expression which tests if
- `value`
- is not zero and is either an odd negative number or an even positive number.

(2 marks)

- e) Give the output produced when the following `main()` function is executed.

```
def main():
    function_ifs(70, 45)

def function_ifs(num1, num2):
    if num1 < num2 and num2 < 60:
        print("A", end = " ")
        if num2 >= 10 and num2 % 2 == 0:
            print("B", end = " ")
        elif num1 % 10 < 3:
            print("C", end = " ")
        print("D", end = " ")
    else:
        if num1 > 50 or num2 < 50:
            print("E", end = " ")
        if num2 % 2 == 1:
            print("F", end = " ")
        print("G", end = " ")

print("H")
```

(2 marks)

- f) Complete the assignment statement in the following code so that the output when the code is executed is: Yes

```
a_string = 
```

```
if len(a_string) >= 5 and a_string[-1] == "s":
    print("Yes")
else:
    print("No")
```

(2 marks)

Question 2 (14 marks)

- a) Using a `for` loop, write the code which prints the following numbers all on one line:

54 51 48 45 42 39 36 33 30 27 24 21 18 15 12 9 6 3 0

(2 marks)

- b) Using a `while` loop, write the code which prints the following numbers all on one line:

54 51 48 45 42 39 36 33 30 27 24 21 18 15 12 9 6 3 0

(2 marks)

- c) Give the output produced when the following `main()` function is executed.

```
def main():
    numbers = [1, 4, 2, 5, 3, 1, 6, 4, 1]
    index = 0
    value = numbers[0]
    total = 0
    while value != 6 and (total == 0 or total % 2 == 1):
        total = total + value
        index = index + 1
        print(str(index) + ".", value)
        value = numbers[index]

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

(3 marks)

- d) Give the output produced by the following code.

```
result = 1
for number in range(6, -1, -1):
    if number > 3:
        if number < 5:
            result = result * number
        else:
            result = result + number
    print(result, end = " ")
print(result)
```

(3 marks)

- e) Complete the output produced by the following code.

```
count_a = 0
count_b = 0
count_c = 0
for num1 in range(6):
    count_a = count_a + 1
    for num2 in range(3):
        count_b = count_b + 1
        count_c = count_c + 1
print("A:", count_a, "B:", count_b, "C:", count_c)
```

A: B: C:

(2 marks)

- f) Complete the output produced by the following code.

```
count_a = 0
count_b = 0
count_c = 0
for num1 in range(2, 5):
    count_a = count_a + 1
    for num2 in range(num1):
        count_b = count_b + 1
count_c = count_c + 1
print("A:", count_a, "B:", count_b, "C:", count_c)
```

A: B: C:

(2 marks)

Question 3 (12 marks)

- a) Complete the output produced when the following `main()` function is executed. Show all your working.

```
def main():
    result = process_letters("BUT")
    print('Result', result)

def process_letters(word):
    new_phrase = ""
    for letter in word:
        result = get_new_letter_index(letter)
        if result == -1:
            return new_phrase
        new_phrase = new_phrase + str(result)
    return new_phrase

def get_new_letter_index(letter):
    characters = "ABCUX"
    position = characters.find(letter)
    return position
```

Result:

(5 marks)

- b) Complete the `get_non_digits()` function which returns a string which is made up of all the characters of the parameter string excluding any characters which are digits (0 to 9). Note that the string which is returned is all in **uppercase** characters. For example, executing the following `main()` function using the completed `get_non_digits()` function gives the output:

1. ABCDB
2. XYPT
3. ABC

Note: in your code you may **NOT** use the `replace()` string method.

```
def main():
    result_word = get_non_digits("ab34CD912B")
    print("1.", result_word)
    print("2.", get_non_digits("12XY3PT"))
    print("3.", get_non_digits("abc"))
```

```
def get_non_digits(word):
```

```
    digits = "0123456789"
```

(5 marks)

- c) In the docstring of the `process_phrase()` function below, add a short description (fifteen words or less) of the function.

```
def process_phrase(phrase):
```

```
    """
```

```
    """
```

(2 marks)

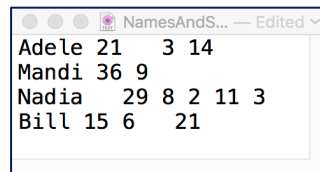
```
    count = 0
    chars = "AEIOU"

    phrase = phrase.upper()
    for letter in phrase:
        if letter in chars:
            count = count + 1
    print(count >= 5)
```

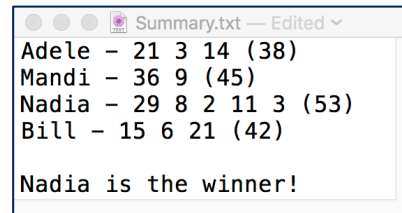
Question 4 (12 marks)

Complete the three functions in the following program which reads information from the input file, 'NameAndScores.txt', processes the information and writes the resulting information to the output file, 'Summary.txt'.

Below is an example of a "NameAndScores.txt" file (on the left) and the corresponding "Summary.txt" file (on the right) produced by the completed program:



```
Adele 21 3 14
Mandi 36 9
Nadia 29 8 2 11 3
Bill 15 6 21
```



```
Adele - 21 3 14 (38)
Mandi - 36 9 (45)
Nadia - 29 8 2 11 3 (53)
Bill - 15 6 21 (42)

Nadia is the winner!
```

NOTE: The `get_players_tuple_list()` function has been completed. This function takes a list of strings as a parameter. Each element of the parameter list is made up of a name followed by two or more scores separated by whitespace, e.g., "Nadia 29 8 2 11 3". The function returns a list of tuples where each element is a tuple with three elements: the name, a string of scores separated by one space and the total of the scores (an integer), e.g., ("Nadia", "29 8 2 11 3", 53).

- Complete the `get_players_info_list()` function which is passed the name of a file as a parameter. Each line of the input file is made up of a name followed by two or more scores. The function returns a list of strings where each element corresponds to one line of the file. No element of the returned list should contain newline characters.
- Complete the `get_winner_name()` function which is passed a list of tuples as a parameter. Each element of the parameter list is a tuple made up of a name, a string of the scores separated by one space and the total of the player's scores (an integer), e.g., ("Nadia", "29 8 2 11 3", 53). This function returns the name of the player who has the highest total score. You can assume that no two players have the same total score.
- Complete the `write_to_file()` function which has three parameters: the output filename, a list of tuples and the name of the winning player. This function firstly writes the information from the list of tuples where each line of the output file contains the name (the first element of each tuple is the name), followed by " - ", followed by the string of scores (the second element of each tuple is the scores string), and finally the total score enclosed in parentheses (the third element of each tuple is the total score integer). Then a blank line is written followed by a line containing the name of the winner and the string "is a winner!". See the screenshot of the example output file above on the right.

```
def main():
```

```
    scores_info = get_players_info_list("NameAndScores.txt")
    players_tuple_list = get_players_tuple_list(scores_info)
    winner = get_winner_name(players_tuple_list)
    write_to_file("Summary.txt", players_tuple_list, winner)
```

```
def get_players_tuple_list(scores_info):
```

```
    #Some of the code for this function is not shown here.
    #This function returns a list where each element is a tuple
    #made up of the player's name, their scores and the total
    #of their scores, e.g., ("Nadia", "29 8 2 11 3", 53)
```


ID:

```
players_tuple_list = []
for player_info in players_list:
    #some of the code is not shown here
    player_tuple = ...
    players_tuple_list.append(player_tuple)
return players_tuple_list
```

```
def get_players_info_list(filename):
```

```
def get_winner_name(players_list):
```

```
def write_to_file(filename, players_tuple_list, winner):
```

```
main()
```

(12 marks)

Question 5 (14 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 = [3, 2, 4, 5]

a_list = a_list + [a_list[-2]]

value = a_list.pop(3)
a_list.append(value - 4)

for number in range(2):
    a_list.insert(2, number)
```

0	1	2	3	4	5	6	7

(4 marks)

- b) Complete the output produced by the following code.

```
list1 = ["a", "b", "c", "d"]
list1 = list1 * 2

index1 = list1.index("c")
list1 = list1[index1:]

print("list1:", list1)
```

<pre>list1: [</pre>	<pre>]</pre>
---------------------	--------------

(4 marks)

- c) Complete the `get_word_from_list()` function which is passed two parameters: a list of single characters and a shift number. The function first reverses the parameter list, and then creates a string from the list elements by translating each element by the shift number. For example, shifting "A" by 3 would result in "D", shifting "Z" by 3 would result in "C", shifting "M" by 5 would result in "R". Executing the following `main()` function gives the output:

```
1: AB
2: GFE
3: KAURI
```

Note: you can assume that all the elements of the parameter list are single uppercase alphabetic characters.

```
def main():
    print("1: ", get_word_from_list(['Y', 'X'], 3))
    print("2: ", get_word_from_list(['A', 'B', 'C'], 4))
    a_list = ['F', 'O', 'R', "X", "H"]
    print("3: ", get_word_from_list(a_list, 3))
def get_word_from_list(letter_list, shift):
```

```
    all_letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

(6 marks)

Question 6 (15 marks).

- a) Complete the following code which prints the total of all the keys of the `a_dict` dictionary which have a seven as the first character of the corresponding string value. The output of the completed code is:

Total: 15

```
a_dict = {2: "7FGST", 4: "4XST", 5: "7TR",
          7: "5ATOR", 8: "7FGY", 9: "4PQY"}
total = 0
```

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

(4 marks)

- b) Complete the output of the following code:

```
a_dict = {1: [4, 2], 2: [3, 4, 2], 4: [3, 1]}
```

```
print("1. a_dict:", a_dict)
```

```
for value_list in a_dict.values():
    for index in range(len(value_list)):
        number = value_list[index]
        if number % 2 == 0:
            value_list[index] = value_list[index] + 1
```

```
print()
```

```
print("2. a_dict:", a_dict)
```

```
1. a_dict: {1: [4, 2], 2: [3, 4, 2], 4: [3, 1]}
```

```
2. a_dict:
```

(4 marks)

ID:

c) Complete the `get_combined_dictionary()` function which is passed two dictionaries, `dict1` and `dict2`, as parameters. The function returns a new dictionary which contains the following:

- all the keys-value pairs of `dict1`,
- all the keys-value pairs of `dict2` but, for any key in `dict2` which is also a key of `dict1` the corresponding string value is the concatenation of the `dict1` corresponding value followed by the `dict2` corresponding value.

For example, executing the following `main()` function using the completed `get_combined_dictionary()` function gives the output:

1. `a_dict1: {1: 'flip', 2: 'bat', 4: 'car', 5: 'dog'}`
2. `a_dict2: {2: 'man', 3: 'cat', 4: 'pet', 7: 'fly'}`
3. `combined_dict: {1: 'flip', 2: 'batman', 4: 'carpet', 5: 'dog', 3: 'cat', 7: 'fly'}`

```
def main():
    a_dict1 = {1: "flip", 2: "bat", 4: "car", 5: "dog"}
    a_dict2 = {2: "man", 3: "cat", 4: "pet", 7: "fly"}
    combined_dict = get_combined_dictionary(a_dict1, a_dict2)
    print("1. a_dict1:", a_dict1)
    print("2. a_dict2:", a_dict2)
    print("3. combined_dict:", combined_dict)
```

```
def get_combined_dictionary (dict1, dict2):
```

(7 marks)

Question 7 (12 marks)

Parts a) and b) of this question refer to two programs which import and use the `tkinter` module. The `main()` functions of the two programs which create the window, create the Canvas object and call the functions for Part a) and Part b) of this question are not shown here.

- a) In the `draw_pattern()` function below, complete the **four** statements marked #1, #2, #3 and #4 so that the output window is as shown lower down on this page. Grid lines have been drawn in the window to help you. The gap between adjacent gridlines is the same as the variable, `size`, i.e., 10 pixels.

```
def draw_pattern(a_canvas):
```

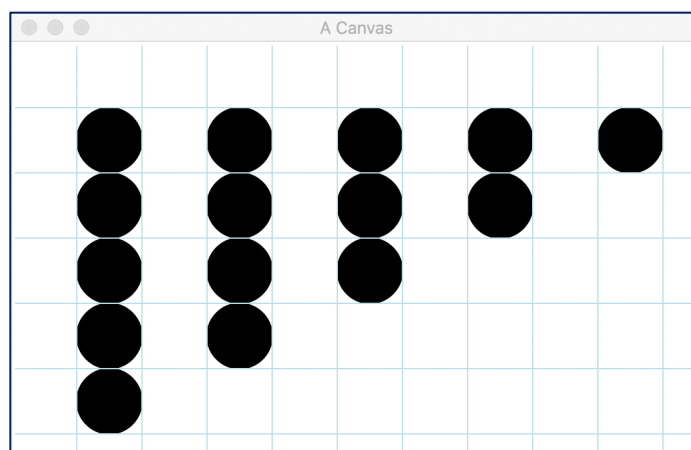
```

    size = 10
    down = size
    for row_number in range(
        ):           # 1.
        left = size
        for col_number in range(
            ):       # 2.
            area = (left, down, left + size, down + size)
            a_canvas.create_oval(area, fill='black')
            left =
                # 3.

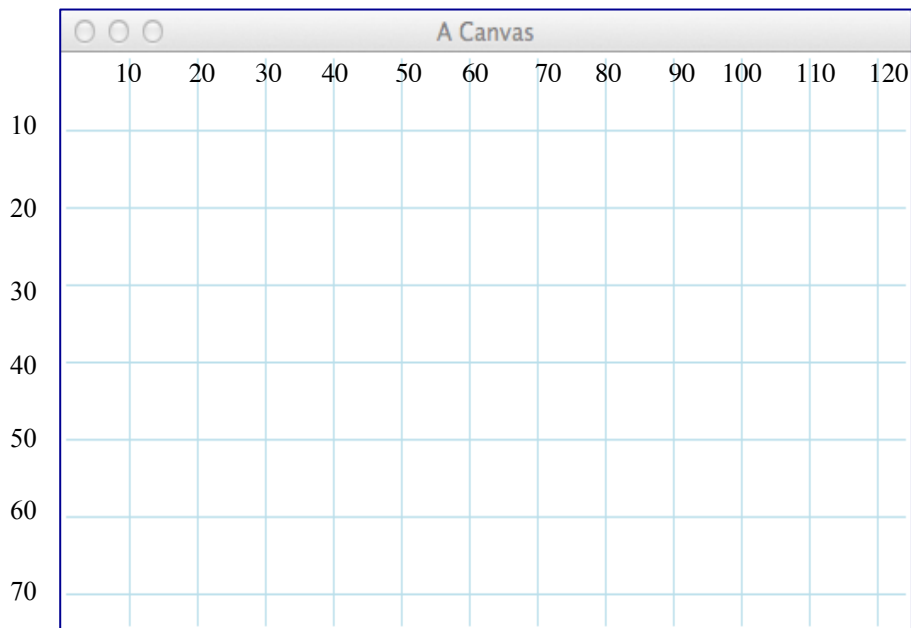
        down =
            # 4.

```

(5 marks)



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



(7 marks)

```
def draw_pattern(a_canvas):
    size = 10
    start_left = size
    pattern_dictionary = {1: "0114220", 2: "0003000", 3: "1110222",
                        4: "0004000"}

    for row_number in pattern_dictionary:
        left = start_left
        down = row_number * size
        for digit in pattern_dictionary[row_number]:
            number = int(digit)
            if number == 1:
                a_canvas.create_line(left + size, down, left,
                                    down + size)
            elif number == 2:
                a_canvas.create_line(left, down, left + size,
                                    down + size)
            elif number == 3:
                area = (left, down, left + size, down + size)
                a_canvas.create_rectangle(area, fill="black")
            elif number == 4:
                area = (left, down, left + size, down + size)
                a_canvas.create_oval(area)
            left = left + size
```

Question 8 (9 marks)

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

```
def main():
    a_list = [3, 4, 1]
    fiddle1(a_list)
    print("a_list:", a_list)

def fiddle1(list1):
    elements_to_add = [5, 5, 3]
    list2 = list1
    for element in elements_to_add:
        if element not in list1:
            list2.append(element)
    list1.pop(1)
```

`a_list:`

(2 marks)

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

```
def main():
    a_list = [3, 5, 7]
    fiddle2(a_list)
    print("a_list:", a_list)

def fiddle2(list1):
    list2 = list1
    list1 = [3, 4]
    list2.reverse()
```

`a_list:`

(2 marks)

- c) In the docstring of the `do_a_check()` function below, add ONE doctest which does not fail.

```
def do_a_check(value1, value2):  
    """Checks the parameter values
```

(2 marks)

```
    """
```

```
    list_of_words = value1.split()  
    return len(list_of_words) == value2
```

```
import doctest  
doctest.testmod()
```

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

```
a_string = "MXQ339"  
a_dict = {"A": "5", "M": [9, 3], "P": "M"}  
a_list = [4, a_dict["P"], 2.5]
```

```
object1 = a_list.index(2.5)  
object2 = a_dict[a_string[0]]  
object3 = a_list[0] * a_list[-1]
```

`object1` is of type:

`object2` is of type:

`object3` is of type:

(3 marks)

OVERFLOW PAGE

(If you have used this page, please indicate clearly under the relevant question that you have overflowed to this page)