THE UNIVERSITY OF AUCKLAND

FIRST SEMESTER, 2018
Campus: City

COMPUTER SCIENCE

Principles of Programming

(Time Allowed: TWO hours)

Note:

- The use of calculators is **NOT** permitted.
- You should separate the Section A Question Booklet from the Section B Question/Answer Booklet. You may keep the Section A Question Booklet. You must hand in the Section B Question/Answer booklet and the Teleform sheet.
- Compare the exam version number on the Teleform sheet supplied with the version number above. If they do not match, ask the supervisor for a new sheet.
- Enter your name and Student ID on the Teleform sheet. Your name and Student ID should be entered left aligned. If your name is longer than the number of boxes provided, truncate it.
- Answer **Section A** on the Teleform answer sheet provided. For Section A, use a dark pencil to mark your answers in the answer boxes on the Teleform sheet. Check that the question number on the sheet corresponds to the question number in this question/answer book. Do not cross out answers on the Teleform sheet if you change your mind. You must completely erase one answer before you choose another one. If you spoil your sheet, ask the supervisor for a replacement. There is one correct answer per question.
- Answer **Section B** in the space provided in the Section B Question/Answer Booklet.
- Attempt all questions. Write as clearly as possible. The space provided will generally be sufficient but is not necessarily an indication of the expected length. Extra space is provided at the end of this exam book.

THIS PAGE HAS BEEN INTENTIONALLY LEFT BLANK.

SECTION A

MULTIPLE CHOICE QUESTIONS

For each question, choose the **best** answer according to the information presented in lectures. Select your preferred answer on the Teleform answer sheet provided by shading in the appropriate box.

Question 1
[2 marks] What is the output of the following code fragment?

```
var1 = 7
var2 = 3
var3 = var1 * var2 - var2
var4 = var2 ** var2 - var3
print(min(max(var1,var3),max(var2,var4)))
```

(a) `18`
(b) `3`
(c) **`9`**
(d) `7`
(e) None of the above.

Question 2
[2 marks] What is the output of the following code fragment?

```
number = 23
result = ""
while number > 0:
    val = number % 2
    number = number // 2
    result = str(val) + result
print(result)
```

(a) **`10111`**
(b) `01000`
(c) `11101`
(d) `00010`
(e) None of the above.

Question 3
[2 marks] What is the output of the following code fragment?

```
text = "the quick brown fox jumps over the lazy dog"
message = text[text.rfind("the") + 4] + text[-17:-14] + \
          text[text.find("brown") - 3] + \
          text[text.rfind("revo s")]
print(message)
```

(a) `zovers`
(b) `q ovog`
(c) `lovecs`
(d) **`lovecg`**
(e) None of the above.

Question 4
[2 marks] What is the output of the following code fragment?

```
def display(message):
    message = message.upper()
    print(message)

message = "Hello World!"
display(message)
print(message)
```

(a) **`HELLO WORLD!`**
`Hello World!`
(b) `Hello World!`
`Hello World!`
(c) `Hello World!`
`HELLO WORLD!`
(d) `HELLO WORLD!`
`HELLO WORLD!`
(e) None of the above.

Question 5
[2 marks] What is the output of the following code fragment?

```
def get_horoscope(number):
    message1 = "Amazing day ahead"
    message2 = "Romance is very likely"
    message3 = "Proceed with caution"
    message4 = "Lucky lucky you"
    if number < 4:
        return message1
    elif number < 7:
        return message2
    elif number < 8:
        return message3
    else:
        return message4

message = get_horoscope(17 % 9)
print(message)
```

(a) `Proceed with caution`
(b) `Romance is very likely`
(c) `Amazing day ahead`
(d) **`Lucky lucky you`**
(e) None of the above.

Question 6
[2 marks] What is the output of the following code fragment?

```
def show_output():
    number = 1
    count = 10
    value = 4
    while count > 4:
        count = count - 2
        value += count
        number += 1
    print(str(number) + ":", count, value)

show_output()
```

(a) `2: 6 12`
(b) **`4: 4 22`**
(c) `1: 8 4`
(d) `3: 4 18`
(e) None of the above.

Question 7
[2 marks] What is the output of the following code fragment?

```
def show_output():
    top = 6
    count = 0
    total = 0

    for bottom in range(0, top + 1, 2):
        count += 1
        total += top + bottom
    print("count:", count, "total:", total)

show_output()
```

(a) `count: 5 total: 45`
(b) `count: 3 total: 24`
(c) **`count: 4 total: 36`**
(d) `count: 3 total: 27`
(e) None of the above.

Question 8
[2 marks] What is the output of the following code fragment?

```
list1 = [7, 3, 3, 9, 6, 4]
list2 = [3, 4, 1, 7, 6]
list3 = list2[2: 4] + [list2[-3]] + list1[-1: -4: -2]
print("list3:", list3)
```

(a) `list3: [1, 7, 1, 7, 6, 4, 9]`
(b) **`list3: [1, 7, 1, 4, 9]`**
(c) `list3: [1, 7, 1, 9, 4]`
(d) `list3: [1, 7, 1, 7, 6, 4, 9, 3]`
(e) `list3: [1, 7, 6, 1, 4, 9]`

Question 9
[2 marks] Given the following code, which of the following correctly states the type of the three variables, `object1`, `object2` and `object3`?

```
a_tuple = (31, 2, 2.0)
a_list = [a_tuple[-1], a_tuple[:2], 'True']
a_dict = {14: [4, 3], 23: [3, 4, 1]}
object1 = a_tuple * 2
object2 = a_list[1]
object3 = a_dict[23] + a_dict[14]
```

(a) `object1: list, object2: float, object3: list`
(b) `object1: tuple, object2: string, object3: list`
(c) `object1: list, object2: tuple, object3: int`
(d) **`object1: tuple, object2: tuple, object3: list`**
(e) None of the above.

Question 10
[2 marks] The `get_result()` function contains five doctests. One of the five doctests fails (i.e., the result returned by the function does not match the expected answer). Which one of the doctest function calls below will fail?

```
def get_result(list_of_numbers):
    """
    >>> get_result([1, 2, 4])
    3
    >>> get_result([])
    0
    >>> get_result([6, 0, 0, 0])
    6
    >>> get_result([6])
    0
    >>> get_result([1, 6])
    6
    """

    if len(list_of_numbers) < 2:
        return 0
    result1 = sum(list_of_numbers) - min(list_of_numbers)
    result2 = len(list_of_numbers) - 1
    result3 = round(result1 / result2)
    return result3
```

(a) **`>>> get_result([6, 0, 0, 0])`**
(b) `>>> get_result([1, 6])`
(c) `>>> get_result([])`
(d) `>>> get_result([6])`
(e) `>>> get_result([1, 2, 4])`

Question 11
[2 marks] Give the output produced when the following `main()` function is executed.

```
def main():
    a_list = [3, 4, 7]
    num = 4
    do_something1(a_list, num)
    print("a_list:", a_list, "num:", num)

def do_something1(list1, num):
    list2 = list1
    num = num + 1
    for count in range(3):
        list2.append(num)
```

(a) `a_list: [3, 4, 7, 5, 5] num: 4`
(b) `a_list: [3, 4, 7] num: 5`
(c) `a_list: [3, 4, 7, 5, 5, 5] num: 5`
(d) `a_list: [3, 4, 7] num: 4`
(e) **`a_list: [3, 4, 7, 5, 5, 5] num: 4`**

Question 12
[2 marks] Give the output produced when the following `main()` function is executed.

```
def main():
    a_list = [3, 7, 5]
    a_tuple = (1, 8)
    do_something2(a_list, a_tuple)
    print("a_list:", a_list, "a_tuple:", a_tuple)

def do_something2(list1, a_tuple):
    list2 = [6, 2, 1]
    a_tuple = (a_tuple[1], a_tuple[0])
    list1 = list2
```

(a) `a_list: [6, 2, 1] a_tuple: (1, 8)`
(b) **`a_list: [3, 7, 5] a_tuple: (1, 8)`**
(c) `a_list: [6, 2, 1] a_tuple: (8, 1)`
(d) `a_list: [3, 7, 5] a_tuple: (8, 1)`
(e) None of the above.

Question 13
[2 marks] Given the following code, how many times are the letters "A", "B" and "C" printed?

```
for num1 in range(5):
    print("A")
    for num2 in range(2):
        print("B")
    print("C")
```

(a) **A: 5 B: 10 C: 5**
(b) A: 6 B: 10 C: 10
(c) A: 4 B: 8 C: 4
(d) A: 5 B: 2 C: 5
(e) A: 6 B: 18 C: 6

Question 14
[2 marks] Given the following code, how many times are the letters "A", "B" and "C" printed?

```
for num1 in range(1, 4):
    print("A")
    for num2 in range(num1):
        print("B")
print("C")
```

(a) A: 3 B: 12 C: 3
(b) A: 4 B: 10 C: 1
(c) A: 4 B: 10 C: 3
(d) **A: 3 B: 6 C: 1**
(e) A: 3 B: 10 C: 3

THE UNIVERSITY OF AUCKLAND

FIRST SEMESTER, 2018
Campus: City

Computer Science

Principles of Programming

(Time Allowed: TWO hours)

SECTION B Question/Answer Booklet

Answer all questions in this section in the space provided. If you run out of space then please use the Overflow Sheet and indicate in the allotted space that you have used the Overflow Sheet.

Surname:	
First Name(s):	
Preferred Name:	
Student ID:	
Login Name (UPI):	

MARKERS ONLY

Q1 – Q14 (/28)	**Q15** (/16)	**Q16** (/16)	**Total** (/100)
Q17 (/14)	**Q18** (/14)	**Q19** (/12)	

Question 15 (16 marks)

a) Complete the `capitalize_each_word()` function below which takes a single string parameter `text`. The function returns a string, where each word in the parameter string has its first letter capitalized. You can assume that words in the parameter `text` are separated by a single space character and that all letters are lowercase. The string returned by the function should be of the exact same length as the parameter `text`.

For example, when the following program is executed with the completed function, the output is:

```
This Is An Easy Exam.
I Am Having Fun!
```

```
def main():
    print(capitalize_each_word("this is an easy exam."))
    print(capitalize_each_word("i am having fun!"))

def capitalize_each_word(text):
```

```
    str_list = text.split()
    new_string = ""
    for string in str_list:
        first_letter = string[0].upper()
        rest_of_word = string[1:]
        new_string += first_letter +
                              rest_of_word + " "
    new_string = new_string.strip()
    return new_string
```

(8 marks)

```
main()
```

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

```
def main():
    a = 5
    print(first(a))

def first(a):
    b = 2
```

```
    for i in range(0, a):
        b += 2
    print("1.",  b)
    return a + b
```

```
1. 12
17
```

(4 marks)

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

```
def main():
    b = 15
    print(second(b))

def second(a):
    b = a * 2
    if a % 2 == 0:
        b = 2
    elif a % 3 == 0:
        b = 3
    elif a % 5 == 0:
        b = 5
    elif a % 7 == 0:
        b = 7
    else:
        b = 11
    print("2.", b)
    return a % b
```

```
2.3
0
```

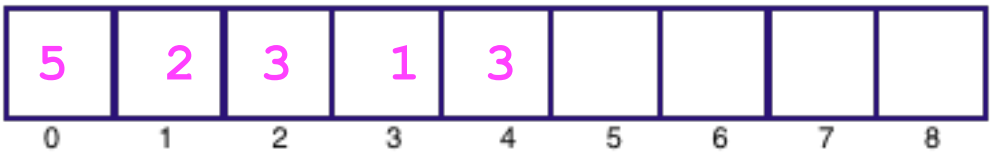
(4 marks)

Question 16 (16 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 = [5, 2, 1, 2, 8]

value = a_list.pop(2)
a_list.insert(2, 3)
a_list.pop()
a_list.insert(3, value)
value = a_list.pop()
value = value + a_list.index(2)
a_list.append(value)
```



(5 marks)

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

```
def main():
    numbers = [4, 3, 3, 2, 1]
    result_list = []
    index1 = 0
    index2 = len(numbers) - 1
    while index1 <= index2:
        num1 = numbers[index1]
        num2 = numbers[index2]
        result_list.append(num1 + num2)
        index1 += 1
        index2 -= 1

    print("Result_list:", result_list)
```

```
Result_list:[5, 5, 6]
```

(5 marks)

c) Complete the `remove_negatives_and_zeros()` function which is passed a list of integers as a parameter. The function removes all the negative numbers and all the zero numbers from the parameter list. For example, executing the following `main()` function using the completed `remove_negatives_and_zeros()` function gives the output:

```
BEFORE: [4, -3, 0, 9, -2, 7, 8, -3, 2, -8]
 AFTER: [4, 9, 7, 8, 2]
```

```
def main():
    a_list = [4, -3, 0, 9, -2, 7, 8, -3, 2, -8]
    print("BEFORE:", a_list)
    remove_negatives_and_zeros(a_list)
    print(" AFTER:", a_list)

def remove_negatives_and_zeros(nums_list):
```

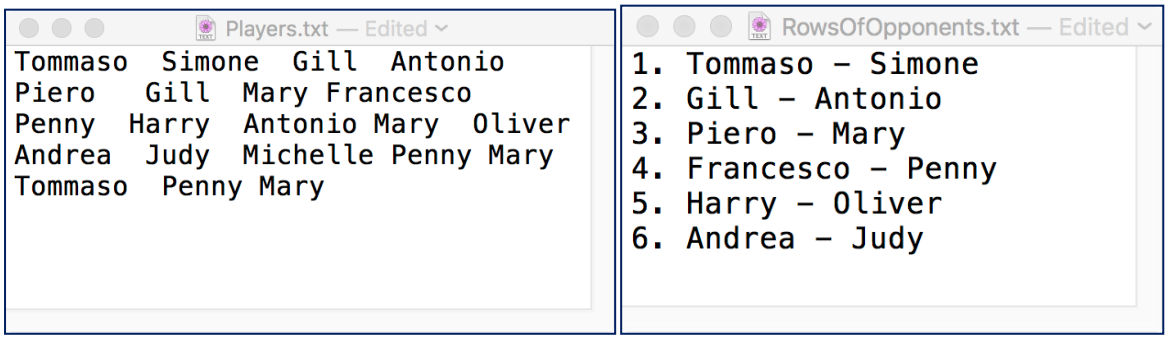
```
for i in range(len(nums_list)-1,-1,-1):
    if nums_list[i] < 1:
        nums_list.pop(i)
```

(6 marks)

Question 17 (14 marks)

The following program reads information from the input file, "`Players.txt`", processes the information and writes output to a text file, "`RowsOfOpponents.txt`". The input file is made up of the names of players each separated by one or more spaces, e.g., "`Tommaso Simone Gill Antonio Piero Gill Mary ...`". Some of the names in the file occur more than once.

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



a) Complete the `get_players_list()` function which is passed one parameter, the name of a file which contains names separated by spaces. The function returns a list of strings where each element corresponds to one name from the file.

b) Complete the `get_uniques_list()` function which is passed a list of strings as a parameter. Each element in the parameter list is made up of a single name. This function returns a new list containing all the **unique** names from the parameter list, i.e., the returned list contains no duplicate strings.

c) Complete the `write_to_file()` function which has two parameters: the name of the output file and a list of unique strings, the names of the players. This function writes all the players' names from the parameter list in order, two players on each line. Each line is numbered (starting from the number 1) and the names of the two players on the line are separated by the string " - ". Note that each number is followed by ". ". See the screenshot of the example output file above on the right.

Note: you can assume that there is always an even number of players in the parameter list of players' names.

```
def main():
        players_list = get_players_list("Players.txt")
        players_list = get_uniques_list(players_list)
        write_to_file("RowsOfOpponents.txt", players_list)
```

```
def get_players_list(filename):
```

```
file_in = open(filename, "r")
contents = file_in.read()
file_in.close()
names_list = contents.split()
return names_list
```

```
def get_uniques_list(names_list):
```

```
uniques_list = []
for name in names_list:
    if name not in uniques_list:
        uniques_list.append(name)
return uniques_list
```

```
def write_to_file(filename, players_list):
```

```
file_out = open(filename, "w")
number = 1
index = 0
while index < len(players_list) - 1:
    line_of_info = str(number) + ". " +
                        players_list[index] + " - "
                    + players_list[index + 1] + "\n"
    file_out.write(line_of_info)
    number += 1
    index += 2

file_out.close()
```

```
main()
```

(14 marks)

Question 18 (14 marks)

a) Complete the following `main()` function which adds 2 to the **last** element of all the value lists corresponding to the keys in the `a_dict` dictionary. You can assume that all the value lists in the dictionary contain at least one element. For example, the output of the completed code is:

```
1. {'a': [3, 4, 1, 6], 'b': [3, 1, 4], 'd': [1, 7], 'c': [5]}
2. {'a': [3, 4, 1, 8], 'b': [3, 1, 6], 'd': [1, 9], 'c': [7]}
```

```
def main():
    a_dict = {"a": [3, 4, 1, 6], "c": [5], "b": [3, 1, 4],
              "d": [1, 7]}
    print("1.", a_dict)

    for num_key in a_dict:
        value = a_dict[num_key]
        value[-1] += 2

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

(4 marks)

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

```
def main():
    word = "CATS"
    result = ""
    a_dict1 = {'A': 4, 'T': 1, 'C': 2, 'R': 3, 'S': 5}
    a_dict2 = {7: 'N', 3: 'T', 2: 'X', 5: 'C', 4: 'E'}
    for letter in word:
        if letter in a_dict1:
            value = a_dict1[letter]
            print(letter, value)
            if value in a_dict2:
                result = result + a_dict2[value]
    print("Result:", result)
```

```
C 2
A 4
T 1
S 5
Result: XEC
```

(4 marks)

c) Complete the `get_dates_dict()` function which is passed a list of strings as a parameter. Each string is made up of a month name, followed by a space followed by a number (the day number), e.g., "May 23". The function returns a dictionary where:

- the keys are the month names,
- the corresponding values are a sorted list of all the **UNIQUE** days (integers) for each month.

Note that the parameter list may contain duplicate strings. Executing the `main()` function below with the completed `get_dates_dict()` function prints:

```
Important dates (April - June):
  April: [2, 3]
  June: [2, 12, 28]
  May: [4, 9, 18, 21, 26, 30]
```

```
def main():
    print("Important dates (April - June):")
    important_dates = ["April 3", "April 3", "June 12",
                       "May 30", "June 2", "May 26",
                      "May 9", "May 18", "June 28",
                       "May 4", "April 2", "May 21", "May 21"]
    a_dict = get_dates_dict(important_dates)
    for month in a_dict:
        print("  ", month, ": ", a_dict[month], sep = "")
```

```
def get_dates_dict(dates_list):
```

```
a_dict = {}
for date_string in dates_list:
   date_list = date_string.split()
   month_name = date_list[0]
   day_number = int(date_list[1])

   if month_name in a_dict:
       if day_number not in a_dict[month_name]:
          a_dict[month_name].append(day_number)
          a_dict[month_name].sort()
   else:
       a_dict[month_name] = [day_number]
return a_dict
```

(6 marks)

Question 19 (12 marks)

For both Parts a) and b) of this question the grid lines have been drawn in the window to help you. The gap between adjacent gridlines is 10 pixels.

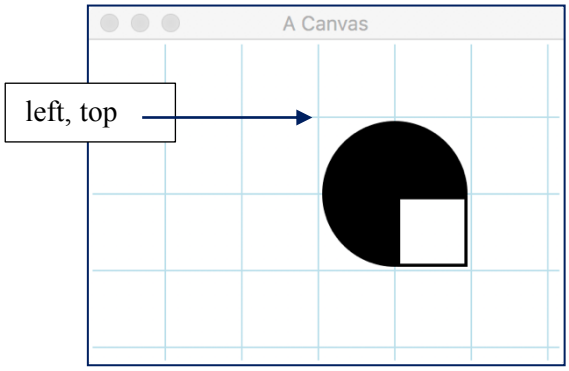
a) Complete the `draw_pattern()` function which draws the following two shapes. Note: your code **MUST** use the parameters `size`, `top` and `left` to draw the shapes.

1. a filled black circle of size 20 pixels (twice `size`), 30 pixels from the left of the window (`left`) and 10 pixels from the top of the window (`top`),

and,

2. a white rectangle of size 10 pixels (`size`) which has the centre of the filled black circle drawn in 1. above as a left top position.

The screenshot on the right shows the output window with the two shapes produced when the following `main()` function is executed.




```
from tkinter import *

def main():
    root = Tk()
    root.title("A Canvas")
    root.geometry("120x70+10+10")
    a_canvas = Canvas(root, bg="white")
    a_canvas.pack(fill=BOTH, expand=1)
    draw_pattern(a_canvas, 30, 10, 10)
    root.mainloop()

def draw_pattern(a_canvas, left, top, size):
```

```
area = (left, top, left + 2 * size, top + 2 * size)
a_canvas.create_oval(area, fill = "black")
area = (left + size, top + size, left + 2 * size,
                                 top + 2 * size)
a_canvas.create_rectangle(area, fill = "white")
```

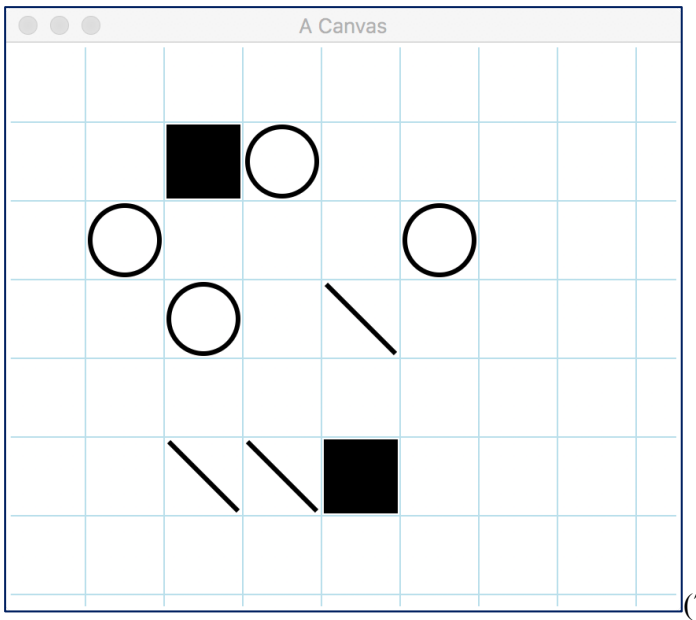
`main()` (5 marks)

b) As accurately as possible, in the window on the next page, show what is drawn when the `main()` function of the following program is executed.

```
from tkinter import *
def draw_pattern(a_canvas):
    size = 10
    pattern_dict = {1: [(30, 50), (20, 50), (40, 30)],
                    2: [(40, 50), (20, 10)],
                    3: [(20, 30), (30, 10), (50, 20), (10, 20)]
                   }
    for shape_type in pattern_dict:
        list_of_points = pattern_dict[shape_type]
        for position_tuple in list_of_points:
            left = position_tuple[0]
            top = position_tuple[1]
            area = (left, top, left + size, top + size)
            if shape_type == 1:
                a_canvas.create_line(left, top, left + size, top + size)
            elif shape_type == 2:
                a_canvas.create_rectangle(area, fill='black')
            elif shape_type == 3:
                a_canvas.create_oval(area)

def main():
    root = Tk()
    root.title("A Canvas")
    root.geometry("125x85+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()
```



(7 marks)