THE UNIVERSITY OF AUCKLAND

FIRST SEMESTER, 2019
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.

SECTION A

MULTIPLE CHOICE QUESTIONS

For each question, choose the **best** answer according to the information presented in lectures.

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

```
value = 3 * 2 ** 3 / (12 - 8) % 13 // 4
print('Result =', value)
```

(a) `Result = 9.0`
(b) `Result = 0.0`
(c) `Result = 1.0`
(d) `Result = 3.0`
(e) None of the above.

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

```
first = 0
second = 1
third = 2
count = 0
print(first,second,third,sep=", ", end=", ")
while count < 3:
    next_val = first + second + third
    print(next_val,", ",sep="", end="")
    first = second
    second = third
    third = next_val
    count = count + 1
```

(a) `0, 1, 2, 3, 5, 7,`
(b) `0, 1, 2, 3, 5, 8,`
(c)
```
0,
1,
2,
3,
6,
11,
```
(d) `0, 1, 2, 3, 6, 11,`
(e) `0, 1, 2, 3, 5, 6,`

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

```
text = "the quick brown fox jumps over the lazy dog"
message = text[(text.rfind("qui") + 6):13] + \
          text[(text.find("evo") - 5):-4] + \
          text[-9:-6] + text[(text.find("jum") - 4)]
print(message)
```

(a) `brozy laf`
(b) `brops over the lazy laf`
(c) `ops over the lazy lax`
(d) `ozy lax`
(e) None of the above.

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

```
def show_output(number):
    if number >= 80:
        print("A", end=' ')
        number = number - 10
    elif number <= 60:
        print("B", end=' ')
        number = number + 20
    elif number % 7 == 0:
        print("C", end=' ')
        number = number - 30
    else:
        print("D", end=' ')
        number = number + 40
    print(number)
show_output(63)
```

(a) `B 83`
(b) `A 53`
(c) `D 103`
(d) `C 33`
(e) None of the above.

Question 5
[2.5 marks] What statement(s) in a while loop structure would change the value of the loop variable(s) so that eventually the loop condition would become false?

(a) Initialization
(b) Body
(c) Condition
(d) Increment
(e) None of the above.

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

```
def show_output():
    for number in range(14, 4, -3):
        number += 3
    print(number, end=' ')
show_output()
```

(a) `14 11 8 5`
(b) `8`
(c) `5`
(d) `4 7 10 14`
(e) None of the above.

Question 7
[2.5 marks] What is the output produced when the following code is executed?

```
a_list = [2, 0, 3, 1]

a_list.insert(2, 4)
a_list.insert(2, 1)

index = a_list.index(4)
a_list.insert(0, index)

a_list.append(a_list[1])

a_list.pop(a_list[1])

print(a_list)
```

(a) `[3, 0, 1, 4, 3, 1, 2]`
(b) `[3, 2, 1, 3, 3, 1, 2]`
(c) `[3, 2, 1, 4, 3, 1, 2]`
(d) `[3, 2, 1, 4, 3, 1, 3]`
(e) None of the above.

Question 8
[2.5 marks] What is the output produced when the following code is executed?

```
a_list1 = [1, 4, 5, 2]
a_list2 = a_list1
a_list3 = a_list1

a_list1.sort()
a_list2.sort()
a_list3.reverse()

print("1: ", a_list1)
print("2: ", a_list2)
print("3: ", a_list3)
```

(a)
```
1:  [5, 4, 2, 1]
2:  [1, 2, 4, 5]
3:  [5, 4, 2, 1]
```
(b)
```
1:  [1, 2, 4, 5]
2:  [1, 2, 4, 5]
3:  [2, 5, 4, 1]
```
(c)
```
1:  [1, 2, 4, 5]
2:  [1, 2, 4, 5]
3:  [5, 4, 2, 1]
```
(d)
```
1:  [5, 4, 2, 1]
2:  [5, 4, 2, 1]
3:  [5, 4, 2, 1]
```
(e) None of the above.

Question 9
[2.5 marks] What is the output produced when the following code is executed?

```
list1 = [3, 4, 7]
list2 = list1
list3 = [3, 4]
list2.pop(2)

print("1:", list1 == list2, " 2:", list1 == list3,\
      " 3:", list1 is list2, " 4:", list1 is list3)
```

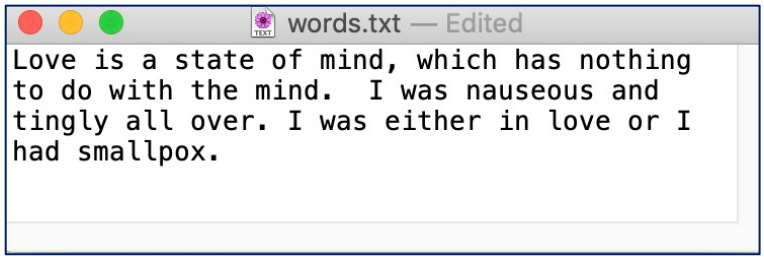
(a) `1: True 2: False 3: True 4: False`
(b) `1: True 2: True 3: True 4: True`
(c) `1: False 2: False 3: False 4: False`
(d) `1: True 2: True 3: True 4: False`
(e) None of the above.

Question 10
[2.5 marks] What is the output produced when the following code is executed?

```
words = "the,lone,wolf,ate,the,juicy,bone"
word_list = words.split(",")
for word in word_list:
    if len(word) == 4:
        index = words.find(word)
        words = words[:index] + words[index + 5:]
print("**" + words + "**")
```

(a) `**the,,,ate,the,juicy,**`
(b) `**lone, wolf, juicy, **`
(c) `**the,ate,the,juicy,**`
(d) `**['the', 'ate', 'the', 'juicy']**`
(e) None of the above.

Question 11
[2.5 marks] Below is a screenshot of the `words.txt` file. What is the output produced when the following code is executed?




```
file_in = open("words.txt", "r")
first_read = file_in.read(5)
second_read = file_in.read(3)
third_read = file_in.read()
file_in.close()
to_print = "**" + third_read[:5] + second_read + "**"
print(to_print)
```

(a) `**Love Lov**`
(b) `**state s a **`
(c) `**states a **`
(d) `**a stais **`
(e) None of the above.

Question 12
[2.5 marks] What is the output produced when the following code is executed?

```
a_dict = {5: 6, 6: 11, 4: 5, 9: 6, 2: 2}
count = 0
for key in a_dict:
    if a_dict[key] in a_dict:
        count = count + 1

print("Count:", count)
```

(a) `Count: 2`
(b) `Count: 3`
(c) `Count: 5`
(d) `Count: 4`
(e) None of the above.

Question 13
[2.5 marks] Given the code below, what is the type of the four variables: `object_1`, `object_2`, `object_3`, `object_4`?

```
object_1 = {"a": 6, "k": 14, "t": 10}

for object_2 in object_1.items():
    object_3 = object_2[0]
    object_4 = object_2[1]
    if object_4 <= 10:
        print(object_2)
```

(a) `object_1: dictionary, object_2: string, object_3: string, object_4: integer`
(b) `object_1: dictionary, object_2: integer, object_3: integer, object_4: string`
(c) `object_1: dictionary, object_2: tuple, object_3: string, object_4: integer`
(d) `object_1: dictionary, object_2: tuple, object_3: string, object_4: string`
(e) None of the above.

Question 14

[2.5 marks] Which of the following code segments will produce this output?

```
012
01
0
```

(a)
```
number_of_rows = 3
for number_to_do in range(number_of_rows, 0, -1):
    for j in range(number_to_do):
        print(number_to_do, end='')
    print()
```

(b)
```
number_of_rows = 3
for number_to_do in range(number_of_rows):
    for j in range(number_to_do):
        print(j, end='')
    print()
```

(c)
```
number_of_rows = 3
for number_to_do in range(number_of_rows):
    for j in range(number_of_rows):
        print(j, end='')
    print()
```

(d)
```
number_of_rows = 3
for number_to_do in range(number_of_rows, -1, -1):
    for j in range(number_to_do):
        print(j, end='')
    print()
```

(e) None of the above.

THE UNIVERSITY OF AUCKLAND

FIRST SEMESTER, 2019
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 an 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 (/35)	**Q15** (/14)	**Q16** (/14)	**Total**
Q17 (/13)	**Q18** (/13)	**Q19:** (/11)	(/100)

Question 15 (14 marks)

a) Complete the `replace_characters()` function below which takes two parameters: a string parameter `text`, and a list of strings `char_list`. All strings in `char_list` will have a length of 1 (i.e., they are single character strings). The `replace_characters()` function replaces all instances of the strings in `char_list` in `text`, with a single `"*"` character . The function then returns the modified `text` string. You can assume that both `text` and `char_list` have a length of at least 1. You can also assume that `char_list` will not contain the string `"*"`. Note, your code **MUST NOT use** the string method `replace()`.

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

```
he*** w*r*d
Exam*tim*,*no*pro*l*m
```

```
def main():
    print(replace_characters("hello world",["o", "l"]))
    print(replace_characters("Exam time, no problem",["b", "e", " "]))
```

```
def replace_characters(text, char_list):
```

```
replacement_char = "*"
for char in text:
    if char in char_list:
        char_pos = text.find(char)
        text = text[:char_pos] + replacement_char + \
        text[char_pos + 1:]
    return text
```

(7 marks)

```
main()
```

b) Using the code tracing technique taught in lectures, complete the code trace of the following program and provide the output.

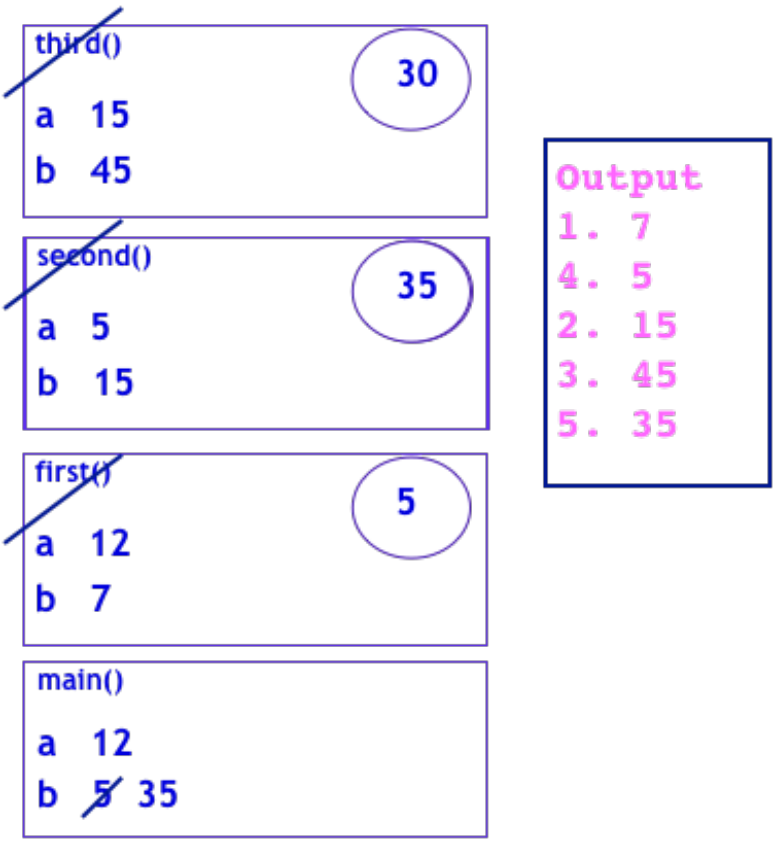
```
def first(a):
    b = a - 5
    print("1.", b)
    return a % b

def second(a):
    b = a + 10
    print("2.", b)
    return a + third(b)

def third(a):
    b = a * 3
    print("3.", b)
    return b - a

def main():
    a = 12
    b = first(a)
    print("4.", b)
    b = second(b)
    print("5.", b)

main()
```



(7 marks)

Question 16 (14 marks)

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

```
def main():
    list1 = [4, 6, 7, 8, 1]
    the_list = [7, 6, 5, 4, 4, 7, 7, 2, 7, 6]
    count = process_lists(list1, the_list)
    print("count:", count, " the_list:", the_list)

def process_lists(list1, list2):
    count = 0
    for element in list1:
        while element in list2:
            index = list2.index(element)
            list2.pop(index)
        count = count + 1
    return count
```

```
count:   5          the_list:   [5, 2]
```

(4 marks)

b) In the `main()` function below, complete the statement which assigns a list of integers to the variable, numbers1, so that the output when the `main()` function is executed is:

```
[72, 5, 36]
```

```
def main():
    numbers1 = [7, 0, 3]
    numbers2 = [2, 5, 6, 4, 1, 8]
    print(concatenate_list_elements(numbers1, numbers2))

def concatenate_list_elements(list1, list2):
    minimum = min(len(list1), len(list2))
    list3 = []
    for index in range(minimum):
        element = str(list1[index]) + str(list2[index])
        element = int(element)
        list3.append(element)

    return list3
```

(4 marks)

c) Complete the `remove_all_e_words()` function which has a list of words (`word_list`) as a parameter. The function removes all the words in the parameter list which contain the lowercase letter 'e'. For example, executing the following `main()` function using the completed `remove_all_e_words()` function gives the output:

```
a_list: ['chip', 'band']
```

```
def main():
    a_list = ["egg", "chip", "peach", "band", "scare", "hem"]
    remove_all_e_words(a_list)
    print("a_list:", a_list)

def remove_all_e_words(word_list):
```

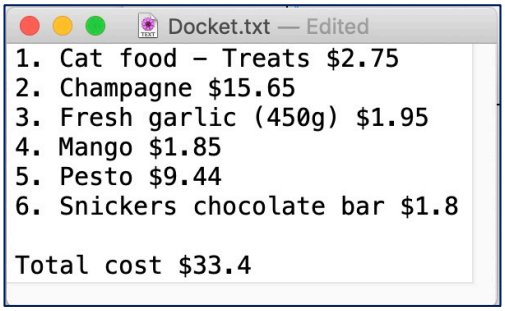
```
    for index in range(len(words_list) - 1, -1, -1):
        element = words_list[index]
        if "e" in element:
            words_list.pop(index)
```

(6 marks)

Question 17 (13 marks)

Complete the three functions in the following program which reads information from the input file, `"Shopping.txt"`, processes the information and writes the resulting information to the output file, `"Docket.txt"`.

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



a) Complete the `get_list_of_items_bought()` function which is passed one parameter, the name of a file which contains a list of the shopping items which have been purchased. In the file, each shopping item is on a separate line. This function returns a list of strings, and each element of the returned list is a string representing each line read from the file. No element of the returned list should contain newline characters.

b) Complete the `get_total_cost()` function which is passed a list of strings as a parameter. Each element of the parameter list is made up of a description of a shopping item followed by a " $" and, finally, the price of the item (a floating point number), e.g., "Cat food - Treats $2.75". This function returns the total of all the item prices in the parameter list rounded to two decimal places.

c) Complete the `write_docket()` function which has three parameters: the name of the output file, a sorted list of strings (each representing one purchased item) and the total price of all the purchased items. This function writes the following information to the output file (see the screenshot of the example output file above on the right):

- a numbered list (starting from the number 1 and followed by `". "`) of each item from the parameter list of strings - one item per line,
- a blank line,
- the final line written to the file is the string `"Total cost $"` followed by the cost parameter.

```
def main():

    cart_list = get_list_of_items_bought("Shopping.txt")
    total_cost = get_total_cost(cart_list)
    cart_list.sort()
    write_docket("Docket.txt", cart_list, total_cost)
```

```
def get_list_of_items_bought(filename):
    file_in = open(filename, "r")
    contents = file_in.read()
    file_in.close()
    contents_list = contents.split("\n")
    return contents_list

def get_total_cost(cart_list):
    total = 0
    for item in cart_list:
        index = item.rfind("$")
        price = float(item[index + 1:])
        total = total + price
    return round(total, 1)

def write_docket(filename, cart_list, cost):
    file_out = open(filename, "w")
    number = 1
    for item in cart_list:
        line = str(number) + ". " + item + "\n"
        file_out.write(line)
        number = number + 1

    file_out.write("\nTotal cost $" + str(cost))
    file_out.close()

main()
```

(13 marks)

Question 18 (13 marks).

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

```
def main():
    a_dict = {4: [6, 9, 4, 5], 5: [8, 5, 6], 7: [5, 6, 9], 14: [5]}
    numbers_list = process_dict(a_dict)
    print("numbers_list:", numbers_list)

def process_dict(a_dict):
    numbers = []
    for key in a_dict:
        for element in a_dict[key]:
            if element not in numbers:
                numbers.append(element)

    numbers.sort()
    return numbers
```

```
numbers_list:  [4, 5, 6, 8, 9]
```

(3 marks)

b) In the `main()` function below, the `a_dict` variable is a dictionary which has single letters as keys and lists of integers as corresponding values. In the `main()` function, add code which changes the corresponding value of each key:value pair in the dictionary from a list of integers to a tuple containing just two elements: the minimum followed by the maximum of the corresponding list of integers. The output when the completed `main()` function is executed is:

```
{'z': (4, 9), 'b': (2, 5), 'p': (5, 9), 'e': (5, 8)}
```

```
def main():
    a_dict = {"z": [6, 9, 4, 5], "e": [8, 5, 6],\
              "p": [5, 6, 9], "b": [5, 2]}

    for key, numbers_list in a_dict.items():
        value = (min(numbers_list),
                 max(numbers_list))
        a_dict[key] = value




    print(a_dict)
```

(4 marks)

c) Complete the `get_list_of_keys()` function which is passed two parameters: a Python dictionary (`a_dict`) and an integer (`number`). The function returns a list of all the keys in the parameter dictionary which have a corresponding list which contains one or more elements which are the same as the `number` parameter. The returned list should be sorted. For example, executing the following `main()` function using the completed `get_list_of_keys()` function gives the output:

```
1. ['April', 'November']
2. ['June', 'May', 'November']
```

```
def main():
    a_dict = {"May": [6, 9, 4, 5], "June": [8, 9, 6],\
              "November": [11, 6, 9], "April": [5, 6, 11]}
    keys1 = get_list_of_keys(a_dict, 11)
    keys2 = get_list_of_keys(a_dict, 9)
    print("1.", keys1)
    print("2.", keys2)
```

```
def get_list_of_keys(a_dict, number):
```

```
list_of_keys = []

for key, value in a_dict.items():
    if number in value:
        list_of_keys.append(key)

list_of_keys.sort()
return list_of_keys
```

(6 marks)

Question 19 (11 marks)

Parts a) and b) of this question refer to two programs which import and use the `tkinter` module. The parts of the `main()` functions of the two programs which create the windows and create the `Canvas` objects of this question, are not shown here.

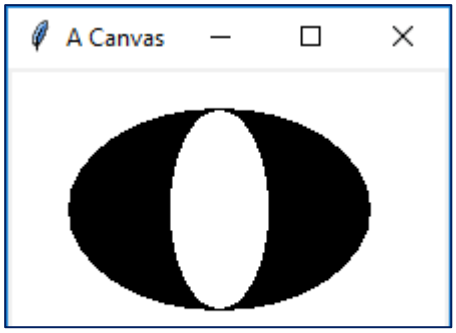
a) Complete the `draw_pattern()` function which draws the following two ovals. Note: your code **MUST** use the parameters `size`, `top` and `left` to draw the ovals.

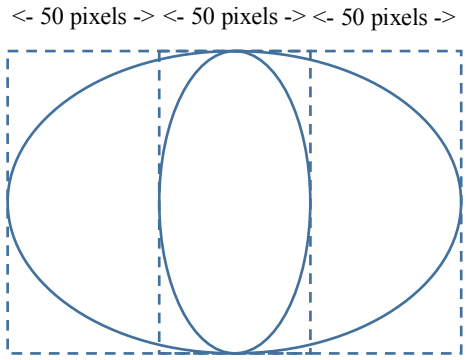
1. a filled black oval of width 150 pixels (three times the value of `size`), height 100 pixels (twice the value of `size`), 30 pixels from the left of the window (`left`) and 20 pixels from the top of the window (`top`),

and,

2. a filled white oval of width 50 pixels (`size`), height 100 pixels (twice the value of `size`), 50 pixels to the right of the filled black circle drawn in 1 above.

The screenshot below on the left shows the output window with the two shapes produced and the diagram on the right shows the dimensions of the two ovals.






```
def main():
    ...
    draw_pattern(a_canvas, 30, 20, 50)
    root.mainloop()

def draw_pattern(a_canvas, left, top, size):
    a_canvas.create_oval(left, top, left + size * 3,
                         top + size * 2, fill = "black")
    a_canvas.create_oval(left + size, top, left + size * 2,
                         top + size * 2, fill = "white")

main()
```

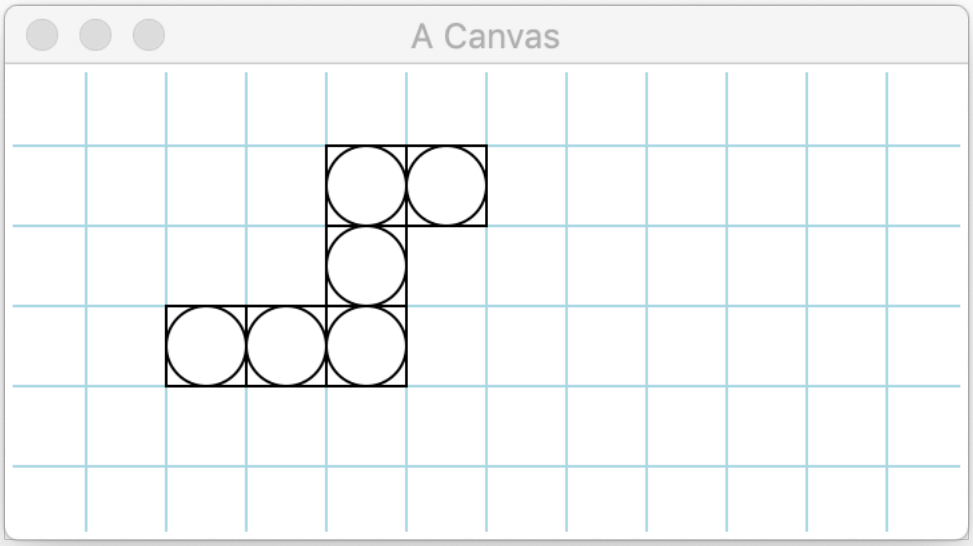
(6 marks)

b) As accurately as possible, in the window below, show what is drawn when the `main()` function of the following program is executed. The grid lines have been drawn in the window to help you. The gap between adjacent gridlines is 10 pixels.

```
def draw_snake(a_canvas):
    left_hand_side = 20
    y_down = 30
    size = 10
    snake_list = [(20,30),(30,30),(40,30),(40,20),(40,10),(50,10)]
    number_of_elements = len(snake_list)
    for number_to_do in range(number_of_elements):
        x_left = snake_list[number_to_do][0]
        y_down = snake_list[number_to_do][1]
        rect = (x_left, y_down, x_left + size, y_down + size)
        a_canvas.create_rectangle(rect)
        a_canvas.create_oval(rect)

def main():
    ...
    draw_snake(a_canvas)
    root.mainloop()

main()
```



(5 marks)