

THE UNIVERSITY OF AUCKLAND

SEMESTER 2, 2015

Campus: City

COMPUTER SCIENCE
TEST SOLUTIONS
 Principles of Programming

(Time Allowed: 75 minutes)

NOTE:

You must answer **all** questions in this test.

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 | |
| Student ID | |
| Login (UPI) | |
| Lab Time | |

| | |
|------------------------|----------------------------|
| Q1 (/39) | Q4 (/14) |
| Q2 (/20) | Q5 (/13) |
| Q3 (/14) | TOTAL (/100) |

ID:

Question 1 (39 marks)

a) Complete the output produced by the following code.

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

Result: 7

(3 marks)

b) Give the output produced by the following code.

```
print(7 % 4, 4 % 4, 5 % 10)
```

3 0 5

(3 marks)

c) Complete the output produced by the following code.

```
result = 1 + 2 * 3 ** 2 // 10
print("Result:", result)
```

Result: 2

(3 marks)

ID:

d) Complete the output produced by the following code.

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

Result: **33545**

(3 marks)

e) Complete the output produced by the following code.

```
word1 = "pastiche"  
word2 = word1[5:] + word1[-1] + word1[2:4]  
print("Letters:", word2)
```

Letters: **cheest**

(3 marks)

ID:

f) Give the output produced by the following code.

```
word1 = "bric a brac"  
word2 = word1.upper()  
word2 = word2.strip()  
position1 = word1.find("b")  
position2 = word1.rfind("B")  
position3 = word2.rfind("B")  
  
print(position1, position2, position3)
```

```
0 -1 7
```

(3 marks)

g) Complete the following Python statement which assigns a random number which is either 8, 10 or 12 to the variable, number. You can assume that the random module has been imported..

```
number = random.randrange(8, 13, 2)
```

(3 marks)

ID:

h) Give the output produced by the following code.

```
num1 = 32
num2 = 150
num3 = 100

if num1 < num2 or num3 > num2:
    print("A")
    if num3 - num2 > 100:
        print("B")
    print("C")
elif num3 < num1:
    print("d")
else:
    if num3 < num2 - 40:
        print("E")
print("F")
```

A
C
F

(3 marks)

ID:

i) Give the output produced by the following code.

```
num1 = 32
num2 = 150
is_a_gift = True
amount = 200

if amount > num1 and amount < num2:
    if is_a_gift and amount < 100:
        amount = amount - 10
        print("1.")
    else:
        amount = amount + 5
        print("2.")
else:
    if amount > num2:
        amount = amount - 20
        print("3.")
    elif amount < num1:
        amount = amount + 20
        is_a_gift = False
        print("4.")
    if is_a_gift:
        amount = amount - 2
        print("5.")

print("Amount:", amount)
```

```
3.
5.
Amount: 178
```

(3 marks)

CONTINUED

ID:

j) Give the output produced by the following code.

```
for number in range(4, 20, 4):  
    print(number)
```

```
4  
8  
12  
16
```

(3 marks)

k) Give the output produced by the following code.

```
for number in range(18, 10, -3):  
    print(number)
```

```
18  
15  
12
```

(3 marks)

ID:

l) Complete the output produced by the following program.

```
def mystery_1(a_list):
    length = len(a_list)
    last_one = a_list[-1]
    count = 0
    for index in range(length - 1):
        if a_list[index] > last_one:
            count += 1
    return count

def main():
    a_list = [24, 12, 30, 18, 26, 14, 32, 18]
    print(mystery_1(a_list))

main()
```

Result: 4

(3 marks)

m) Given the following code (the right hand side of the first statement is not shown but you can assume the at the code executes without errors):

```
thing1 = ...
thing2 = thing1 + [5, 7, 2]
thing2[1] = "Two"
```

what is the type of the two Python objects `thing1` and `thing2[1]`?

`thing1` is an object of type: `list`

`thing2[1]` is an object of type: `str`

(3 marks)

CONTINUED

ID:

Question 2 (20 marks)

Part a) and Part b) refer to the following function:

```
def process(word1, word2, how_many):  
    part1 = word1[0:how_many]  
    part2 = word2[0:how_many]  
    if word1 == word2:  
        return False  
    else:  
        return part1 == part2  
print("F")
```

a) Complete the call to the process () function so that the following code prints "Yes".

```
if process( "pastiche", "pastry", 4 ):  
    print("Yes")  
else:  
    print("No")
```

(3 marks)

b) Complete the call to the process () function so that the following code prints "No".

```
if process( "pastiche", "cake", 4 ):  
    print("Yes")  
else:  
    print("No")
```

(3 marks)

ID:

- c) Complete the `has_same_ending()` function which is passed two string parameters, `word1` and `word2`. The function returns `True` if both parameter strings have the same last letter, otherwise the function returns `False`. You can assume that the parameter strings always contain at least one character. For example, the following code:

```
print(has_same_ending("taro", "carrot"))
print(has_same_ending("funny", "day"))
```

prints:

`False`

`True`

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

```
    word1_last = word1[-1]
    word2_last = word2[-1]
    return word2_last == word1_last
```

(7 marks)

ID:

- d) Complete the `get_hollow_word()` function which is passed a string parameter, `word`. The function returns a string made up of the first letter of the parameter string followed by a series of "-" symbols followed by the last letter of the parameter string. The string returned by the function has the same number of characters as the parameter string. You can assume that the parameter string always contains at least two characters.

For example, the following code:

```
print(get_hollow_word("Geraldine"))
print(get_hollow_word("Simon"))
```

prints:

```
G-----e
```

```
S---n
```

```
def get_hollow_word(word):
```

```
    first = word[0]
    last = word[-1]
    middle_length = len(word) - 2
    return first + "-" * middle_length +
                                   last
```

(7 marks)

ID:

Question 3 (14 marks)

- a) Convert the following code which uses a `while` loop into equivalent code which uses a `for ... in range(...)` loop.

```
num = 2
while num < 34:
    print(num)
    num = num + 3
```

```
for num in range(2, 34, 3):
    print(num)
```

(6 marks)

ID:

- b) Complete the following program which continuously prompts the user for a whole number (using the prompt "Enter number: ") until the sum of the numbers entered by the user is greater than 20. Once the sum is greater than 20 the final line of output displays the string "Final sum: " followed by the final sum reached (a number greater than 20). An example execution of the completed program is shown below (the user input is shown in bold in a larger font size):

```
Enter number: 9
Enter number: 4
Enter number: 11
Final sum: 24
```

```
def main():
```

```
    user_num = 0
    total = 0

    while total <= 20:
        user_num = int(input("Enter number: "))
        total = total + user_num

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

(8 marks)

```
main()
```

ID:

Question 4 (14 marks)

a) Complete the output produced by the following code:

```
a_list = [3, 2, 1, 5, 0, 3]
a_list[3] = a_list[2] + a_list[1]
a_list[1] = a_list[5] * a_list[0]
a_list[4] = a_list[a_list[2]]
print("a_list:", a_list)
```

```
a_list: [3, 9, 1, 3, 9, 3]
```

(6 marks)

ID:

- c) Complete the `get_count_negatives()` function, which returns the number of elements from the parameter list which are less than zero. For example, executing the following code using the completed function:

```
a_list = [3, -2, 1, 5, 10, -3, 0, -2]
count = get_count_negatives(a_list)
print("Elements less than zero:", count)
```

gives the output:

```
Elements less than zero: 3
```

```
def get_count_negatives(a_list):
```

```
    count = 0

    for num in a_list:
        if num < 0:
            count += 1

    return count
```

(8 marks)

ID:

Question 5 (13 marks)

- a) Using the code tracing technique shown in lectures, perform a code trace for the following program and give the output. Give the output in the space below and **show the code trace in the space provided on the next page.**

```
def main():
    print("A")
    partA = "XYZ"
    partB = 123
    function1(partA, partB)
    print("B")

def function1(part1, part2):
    print("C")
    combined = part1 + str(part2)
    print("D", combined)
    result = function2(combined, "Z")
    print("E", result)

def function2(symbols, letter):
    pos = symbols.find(letter)
    print("F", pos)
    result = symbols[pos:] + symbols[pos - 1]
    return result

main()
```

Give the output:

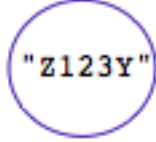
```
A
C
D XYZ123
F 2
E Z123Y
B
```

(6 marks)

CONTINUED

ID:

Show the code trace in the space below:

| | |
|--|---|
| <pre>function2() function symbols "XYZ123" letter "z" pos 2 result "z123Y"</pre> |  |
| <pre>function1() function part1 "XYZ" part2 123 combined "XYZ123" result "z123Y"</pre> | |
| <pre>main() function partA "XYZ" partB 123</pre> | |

(7 marks)

CONTINUED