THE UNIVERSITY OF AUCKLAND

Semester Two, 2016
Campus: City

TEST

COMPUTER SCIENCE

Principles of Programming

(Time Allowed: 75 Minutes)

NOTE:
You must answer **all** questions in this test.
No calculators or smart watches 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	
Username	
Lab Time	

Q1 (/20)	**Q4** (/20)
Q2 (/20)	**Q5** (/20)
Q3 (/20)	**TOTAL** (/100)

Question 1 (20 marks)

a) Complete the output produced by the following code.

```
a = 2 % 5
b = 10 % 5
c = 12 % 9

print("a:", a)
print("b:", b)
print("c:", c)
```

```
a: 2
b: 0
c: 3
```

(3 marks)

b) Complete the output produced by the following code.

```
a = 6
b = a
a = b + 3
c = 4
c = c + 2
d = c
d = d * 2

print("a:", a)
print("b:", b)
print("c:", c)
print("d:", d)
```

```
a:  9
b:  6
c:  6
d:  12
```

(4 marks)

c) Complete the output produced by the following code.

```
amount = 5 + 2 * 3 ** 2 - 2
print("Amount:", amount)
```

```
Amount: 21
```

(3 marks)

d) Complete the following program which prints the cost of international cellphone calls which are priced in the following way:

- every call has a base cost of $2.50.
- the first five minutes of the call are free. Every minute over five minutes costs 23 cents.

The program prompts the user for the number of minutes using the prompt, `"Minutes: "`, calculates the cost based on the above information and prints the cost rounded to two decimal places, followed by the total number of minutes within parentheses. For example, the following screenshots show four different executions of the completed program:

```
Minutes: 4
$2.5 (Minutes: 4)
```

```
Minutes: 6
$2.73 (Minutes: 6)
```

```
Minutes: 1
$2.5 (Minutes: 1)
```

```
Minutes: 15
$4.8 (Minutes: 15)
```

```
base_price = 2.5
minutes_free = 5
rate_per_minute = 0.23

minutes = int(input("Minutes: "))
minutes_to_pay = (minutes - minutes_free)
minutes_to_pay = max(minutes_to_pay, 0)

cost = round(base_price + rate_per_minute *
                                    minutes_to_pay, 2)

print("$", cost, " (Minutes: ", minutes, ")", sep
= "")
```

(10 marks)

Question 2 (20 marks)

a) Assume that the variable, `value`, has been initialised. Write a boolean expression which tests if `value` is an odd number between 5000 and 6000.

```
value % 2 == 1 and value > 5000 and value < 6000
```

(3 marks)

b) In the `main()` function below, complete the statement which assigns a number to the `value` variable, so that the program prints `"That is a maybe!"`

```
def main():
    value = 11 #odd number greater than 10
    if value % 2 == 0 and value > 4:
        print("That is right!")
    elif value <= 4 or value == 12:
        print("That is wrong!")
    elif value > 10:
        print("That is a maybe!")

main()
```

(3 marks)

c) Complete the `get_last_word()` function which is passed one parameter, a string which is made up of two or more words separated by spaces. The function returns the last word in the parameter string. You can assume that the parameter string always contains at least one space character. For example, executing the following `main()` function with the completed function, prints:

```
1. mud
2. tide
3. floss
```

```
def main():
    print("1.", get_last_word("happy as a pig in mud"))
    print("2.", get_last_word("happy as a clam at high tide"))
    print("3.", get_last_word("mental floss"))
```

```
def get_last_word(words):
```

```
position = words.rfind(" ")
return words[position + 1: ]
```

(7 marks)

d) Complete the `check_letters()` function which is passed three parameters, two strings and an integer index value. The function gets the character at the position given by the index value from both the parameter strings and returns `True` if the characters are the same, otherwise the function returns `False`. For example, executing the following `main()` function with the completed function, prints:

```
1. True
2. True
3. False
```

```
def main():
    print("1.", check_letters("think","twice", 2))
    print("2.", check_letters("mumbo","jumbo", 3))
    print("3.", check_letters("mumbo","jumbo", 0))
```

```
def check_letters(word1, word2, index_to_test):
```

```
letter1 = word1[index_to_check]
letter2 = word2[index_to_check]
return letter1 == letter2
```

(7 marks)

Question 3 (20 marks)

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

```
2 5 8 11 14
```

```
for num in range(2, 15, 3):
    print(num, end=" ")
```

(5 marks)

b) Give the output produced by the following code.

```
count = 1
num1 = 3
num2 = 13

while num1 < num2:
    num1 = num1 + 2
    print(count, "-",  num1, num2)
    num2 = num2 - 2
    count = count + 1

print("Finally", num1, num2)
```

```
1 - 5 13
2 - 7 11
3 - 9 9
Finally 9 7
```

(7 marks)

c) Rewrite the following code using an equivalent `for ... in range()` loop instead of the `while` loop:

```
number = 3
count = 10
while count >= 0:
    print(number)
    number = number + count
    count = count - 2

print("Final", number)
```

```
number = 3
for count in range(10, -1, -2):
   print(number)
   number = number + count

print("Finally", number)
```

(8 marks)

Question 4 (20 marks)

a) Given the following code:

```
a_list = ['a', 'b', 'cc', '5']

object1 = a_list[3] + "2"
object2 = a_list + ["Huh!"]
object3 = len(a_list[2])
```

what is the type of each of the three Python objects: `object1`, `object2` and `object3`?

```
object1 is of type: str

object2 is of type: list

object3 is of type: int
```

(6 marks)

b) Using the code trace technique taught in lectures, perform a code trace on the following program and show the output.

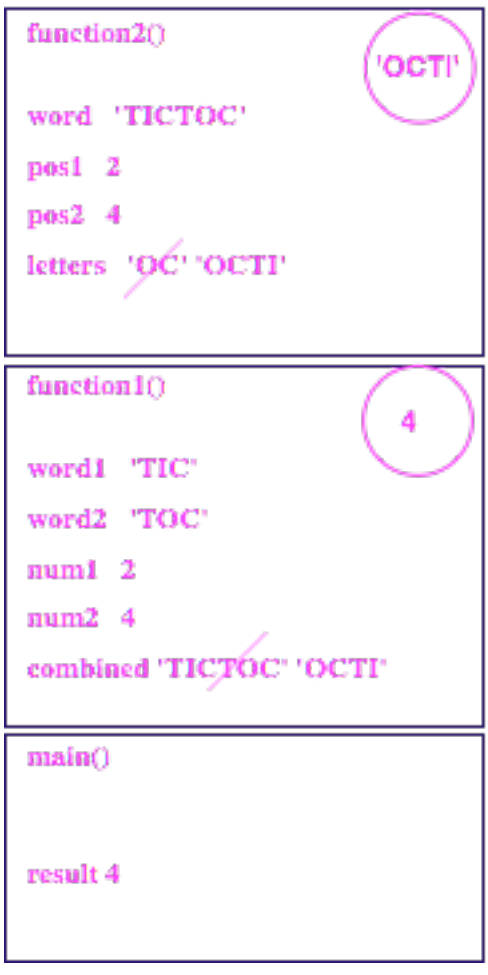
```
def function1(word1, word2, num1, num2):
    print("A.")
    combined = word1 + word2
    combined = function2(combined, num1, num2)
    print("B.", combined)
    return len(combined)

def function2(word, pos1, pos2):
    letters = word[pos2:]
    print("C.", letters)
    letters = letters + word[:pos1]
    return letters

def main():
    result = function1("TIC", "TOC", 2, 4)
    print("D.", result)
main()
```

Output

A.
C. OC
B. OCTI
D. 4



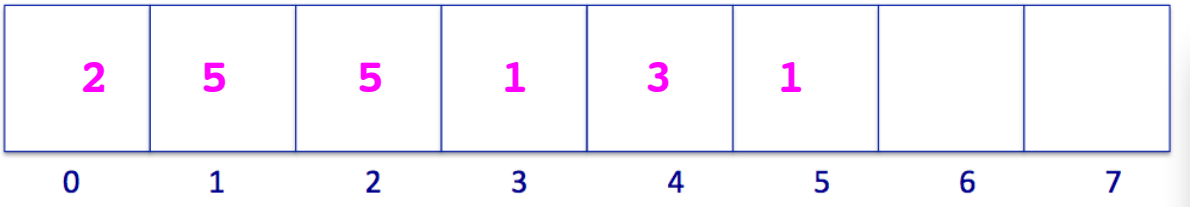
(14 marks)

Question 5 (20 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, 4, 3, 1]

a_list[1] = a_list[1] + a_list[3]
a_list[2] = a_list[3] * a_list[1]
a_list = [a_list[3] - a_list[4]] + a_list
a_list[3] = a_list[-1]
```



(5 marks)

b) Complete the output produced by the following code:

```
a_list = [4, 7, 3, 1, 6, 5]
amount = 20
for num in a_list:
    if num % 2 == 0:
        amount = amount - num

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

Final amount: 10

(4 marks)

c) Complete the `print_count()` function which is passed a list of names as a parameter. The function prints the count of all the names in the names list which have the letter 'i' as their second letter. Note that the function should execute correctly for any list of names. For example, executing the following program with the completed function, prints:

```
Number of names with 'i' as their 2nd letter: 2
```

Note: You can assume that all the elements in the list contain two or more letters.

```
def print_count(names_list):
    text = "Number of names with 'i' as their 2nd letter:"
    count = 0
    for name in names_list:
        if name[1] == 'i':
            count = count + 1

    print(text, count)
```

(11 marks)

```
def main():
    names = ["Cody", "Baldassar", "Delilah", "Vinnie", "Leila",\
        "Zac", "Aiden", "Zaynab", "Maizie", "Maia", "Fede"]
    print_count(names)

main()
```