THE AUSTRALIAN NATIONAL UNIVERSITY

Mid Examination – March 2018

COMP1730 / COMP6730

Programming for Scientists

Study Period: 15 minutes

Time Allowed: 2 hours

Permitted Materials: One A4 page (1 sheet) with notes on both sides. NO calculator permitted. Use a blue or black pen.

Questions are NOT equally weighted. The questions and each of their parts are NOT in order from easy to hard.

***Question 6 is for "COMP6730 students only"**. This question should only be answered by students enrolled in COMP6730. Students in COMP1730 will not receive any marks for answering this question.*

*For **COMP1730 students**, the exam will be marked out of 20, and is worth 20% of the final course mark.*
*For **COMP6730 students**, the exam will be marked out of 24, and is worth 20% of the final course mark.*

Where not otherwise indicated, questions are followed by framed blank panels into which your answers are to be written. If the space in the answer panel is not enough, use the additional panels at the end of the exam paper. If you use an additional panel, make sure you specify which question and part it is for.

Please write and express yourself clearly – if we cannot read what you have written or understand what you have tried to say, your answer will not be considered correct.

Student Number (NOT your name):	Course (write "1730" or "6730"):

The following are for use by the examiners.

Q1 (5)	Q2 (4)	Q3 (3)	Q4 (4)	Q5 (4)	Q6 (0/4)	Total (20/24)

Student Number: .....

Question 1 [5 marks]

(a) Which of the following names are valid function names in Python? (Write your answer, 'yes' or 'no', in the table.)

Name	Yes or No?	Name	Yes or No?
`Else`		`_while`	
`class`		`n`	
`What_am_I?`		`list`	
`3rd_position`		`FALSE`	

[1 mark]

(b) For each of the following Python 3 expressions, write the value that it evaluates to, and the type of that value, into the table. If evaluating the expression results in a runtime error, you only have to write "error".

Expression	value	type
`27 // 5`		
`int(7.5) <= int(7)`		
`15 / 3 + 3`		
`-2 ** 2 * 2`		
`float("3" * 3)`		
`-27 % 5`		
`int(2.5) * '5 == 10'`		
`'1' < '30' >= '5'`		
`list('abc') + ["a"]`		

[3 marks]

(c) It is usually a bad idea to use the equality operator `==` to compare two floating point numbers. Explain how we should compare floating point numbers instead. You may write sample code to help with your explanation but you are not required to do so.

[1 mark]

Student Number: .....

Question 2 [4 marks]

(a) A function `fun_a` is defined as follows:

```
def fun_a(x, y):
    total = 0
    while x - y > 0:
        total = total + 1 / x
        x = x - 1
    return total
```

For each of the following calls to the function, answer whether the function will return a value, enter an infinite loop, or if a runtime error will occur. You do not have to calculate the exact value that the function returns or name the specific error.

(*i*) `fun_a(10, 5)`

(*ii*) `fun_a(5, -1)`

[2 marks]

(b) Two functions `fun_b` and `fun_c` are defined as follows:

```
def fun_b(x):
    print('Fun_b:', x * 3)
    return -x

def fun_c(x):
    print('Fun_c:', fun_b(x + 1))
    return fun_b(x)
```

Write everything that is printed by the call: `print(fun_b(2) + fun_c(4))`
Note you must write the output in order, and make it clear where there are line breaks.

[2 marks]

Question 3 [3 marks]

Each of the following pieces of Python code attempts to add up the integers from 1 through to `n` (inclusive). For each one, answer whether it is correct, meaning that it *runs without error and prints the correct value*. If it is not, explain *precisely* what is wrong with it. (For example, if it has a syntax error, describe which line, or part of a line, is incorrect; if it runs but prints the wrong value, give a concrete example of inputs and output.) Note that any number (zero, one, two or three) of them may be correct. Assume that variable `n` has been set and is a positive `integer` value.

(a)
```
def triangle(n):
    total = 0
    x = 0
    while x <= n:
        x = x + 1
        total = total + x
    return total

print("The sum of the numbers from 1 to n is:", triangle(n))
```

(b)
```
def triangle(n):
    total = 0
    while total <= n:
        total = total + 1
    return total

print("The sum of the numbers from 1 to n is:", triangle(n))
```

(c)
```
def triangle(n):
    if n == 0:
        return 0
    else:
        return triangle(n - 1) + n

print("The sum of the numbers from 1 to n is:", triangle(n))
```

Question 4 [4 marks]

(a) Suppose `s1 = 'To be, or not to be'` and `s2 = 'Et tu, Brute?'` For each of the following expressions, write the value that it evaluates to into the table. If it evaluates to a `string`, remember to put the value in quotation marks. If evaluating the expression results in an error, you only have to write "error".

Expression	value	Expression	value
`s1[4] + s2[4]`		`s2[-3]`	
`s1[-len(s2)] + s2[-len(s1)]`		`s1[3:7]`	
`s2[-8:8]`		`s1[-8:8:-1]`	
`s1[len(s2):]  + s2[len(s1):]`		`s2[4::-1]`	

[2 marks]

(b) Write a function `count_vowels(input_string):` that returns the number of vowels in the provided input string. Your function should count vowels of both upper and lower case.

[1 mark]

(c) What is printed when the following code is run:

```
output = ''
input = [72, 97, 114, 100, 32, 69, 120, 97, 109]
for element in input:
    output = output + chr(element)
print(output)
```

Note that `ord('A')=65`, `ord('a')=97` and `ord(' ')=32`.

[1 mark]

Question 5 [4 marks]

Consider the following function definition:

```
def a_function(a_parameter):
    b = len(a_parameter) - 1
    while b >= 0:
        if a_parameter[b] < 0:
            a_parameter[b] = a_parameter[b] + 1
        b = b - 1
    return a_parameter
```

(a) What is returned by the function call: `a_function([5, 9, -3, -1, 7, 4])`
Note you must clearly indicate the `type` of the returned value as well:

[1 mark]

(b) Explain in plain English what this function does *in general*. Make your explanation as general and informative as you can. A good answer is one that describes the purpose of the function – something you might put into a docstring – *not* a line by line description of how it works.

[1 mark]

(c) What requirements exist on the function argument `a_parameter`, including possible type(s), possible value(s), etc., in order for the function to run without an error. Be as general as possible.

[1 mark]

(d) Re-write the definition of `a_function` so it does the same thing *without* using any sort of loop (`for`, `while` or `list` comprehension).

[1 mark]

Student Number: .....

Question 6 [COMP6730 Only: 4 marks]

This question is for COMP6730 students only. If you are a COMP1730 student you will not receive any marks for attempting this question.

(a) Consider the following function:

```
def fun_d(x):
    while x + 1 > 1:
        x = x / 2
        y = -int(x)
        while y != 0:
            y = y - 1
    return x + y
```

For each of the following function calls, determine whether the function returns a value, gets stuck in an infinite loop or results in a runtime error, and *explain* why this happens. You do not need to determine the value returned or the specific error type.

(*i*) `fun_d(5)`

(*ii*) `fun_d(0.5)`

[2 marks]

(b) Assume we have declared the following variables:

```
l1 = ['what', 'light', 'through', 'yonder', 'window', 'breaks?']
s1 = 'Once more unto the breach'
```

For each of the following expressions, write the value that the expression evaluates to. Make sure to clearly indicate the `type` of the value. If evaluating the expression results in a runtime error, just write "error".

Expression	**value**
`l1[-4:4]`	
`s1[5:10] * s1.find('e')`	
`l1[-2::-3] + list(l1[-1][-3:])`	
`s1[2::2].title()`	

[2 marks]

Student Number: .....