Duration: **110 minutes**
Aids Allowed: **none**

Student Number: |_|_|_|_|_|_|_|_|_|_|_|

Family Name(s): _____

Given Name(s): _____

Lecture Section: ☐ LEC01 (with Michael Guerzhoy)

☐ LEC02 (with François Pitt)

---

*Do **not** turn this page until you have received the signal to start.*
In the meantime, please read the instructions below *carefully*.

This term test consists of 6 questions on 20 pages (including this one), printed on both sides of the paper. *When you receive the signal to start, please make sure that your copy of the test is complete, fill in the identification section above, write your student number where indicated at the bottom of every odd-numbered page (except page 1), and write your name on the back of the last page.*

Answer each question directly on the test paper, in the space provided, and use the reverse side of the pages for rough work. If you need more space for one of your solutions, use the reverse side of a page and *indicate clearly the part of your work that should be marked.*

When you are asked to write code, *no* error checking is required: you may assume that all user input and argument values are valid, except where explicitly indicated otherwise.

Write up your solutions carefully! Comments and docstrings are *not* required, except where explicitly indicated otherwise. However, they may help us mark your answers, and part marks *will* be given for showing that you know the general structure of an answer, even if your solution is incomplete.

If you are unable to answer a question (or part), you will get 20% of the marks for that question (or part) if you write "I don't know" and nothing else—you will *not* get those marks if your answer is completely blank, or if it contains contradictory statements (such as "I don't know" followed or preceded by parts of a solution that have not been crossed off).

MARKING GUIDE

1: _____/15

2: _____/ 8

3: _____/ 8

4: _____/ 5

5: _____/ 6

6: _____/ 8

TOTAL: _____/50

Good Luck!

OVER...

Use this page for rough work—clearly indicate any section(s) to be marked.

Question 1. [15 MARKS]

Each of these subquestions contains a block of code. Treat each block of code independently (*i.e.*, code in one question is not related to code in another), and answer each question in the space provided.

Part (a) Simple Syntax [1 MARK]

```
def add_ints(x, y):
    return x + y

x = 1
y = 2
add_ints()
```

In the block of code to the left, circle every line that would cause the code to fail—there is at least one. Then, in the space below, explain why the line(s) fail.

Part (b) Simple Syntax [1 MARK]

```
def loopy(L)
    for item in L
        print item

my_list = [1, 2, 3]
loopy(my_list)
```

In the block of code to the left, circle every line that would cause the code to fail—there is at least one. Then, in the space below, explain why the line(s) fail.

Part (c) Scope [1 MARK]

```
def sum_to_n(n):
    total = (n * (n + 1)) / 2.0

sum_to_n(5)
print total
```

In the block of code to the left, circle every line that would cause the code to fail—there is at least one. Then, in the space below, explain why the line(s) fail.

Use this page for rough work—clearly indicate any section(s) to be marked.

Question 1. (CONTINUED)

Part (d) Scope [1 MARK]

```
value = 1

def printer(value):
    print value
    value = 42
    print value

print value
printer(value)
print value
```

What is the output of the code to the left?

Part (e) Order of Execution [1 MARK]

```
def printer():
    print "Hello"
print "Hi"

printer()
printer()
```

What is the output of the code to the left?

Part (f) Order of Execution [1 MARK]

```
var_A = 11
var_B = var_A
var_A = 42
```

After this code is executed, the value of `var_B` is:

_____

Use this page for rough work—clearly indicate any section(s) to be marked.

Question 1. (CONTINUED)

Part (g) While Loops [1 MARK]

```
def find_o(search_str):
    index = 0
    while index < len(search_str) and \
            search_str[index] != 'o':
        index += 1
    return index

print find_o("eeyore")
print find_o("pooh")
print find_o("tigger")
```

What is the output of the code to the left?

Part (h) Mutability [1 MARK]

```
def doubler(L):
    for item in L:
        item = item * 2
    print L

my_list = [1, 2, 3]
doubler(my_list)
```

What is the output of the code to the left?

Part (i) Aliasing and Mutability [1 MARK]

```
def doubler(L):
    dL = L
    for index in range(len(dL)):
        dL[index] = dL[index] * 2

my_list = [1, 2, 3]
doubler(my_list)
print my_list
```

What is the output of the code to the left?

Use this page for rough work—clearly indicate any section(s) to be marked.

Question 1. (CONTINUED)

Part (j) Conditionals and Booleans [2 MARKS]

The table to the right shows how an employee's age and experience affects his or her hourly wage. Assume that you have a `boolean` variable `experienced` and an `int` variable `age` that correspond with the labels in the table. Fill in the conditions in the code below to calculate the hourly wage for the employee.

	Experienced?	
Age	Yes	No
under 18	$12.00	$9.50
18 and over	$15.00	$10.50

```
if ____________________________________________________________________________:

    if ____________________________________________________________________:
        wage = 12
    else:
        wage = 15
else:
    if ____________________________________________________________________:
        wage = 9.5
    else:
        wage = 10
```

Part (k) Data Types [2 MARKS]

Fill in the blank so that when this code is run, the user is asked to enter two numbers and then the average of those numbers is printed. The payrates are likely to contain decimal values.

```
num1 = raw_input("Please enter your hourly wage: ")
num2 = raw_input("Please enter your friend's hourly wage: ")


print "Your average wage is", ________________________________________________
```

Part (l) Calling Functions [2 MARKS]

Fill in the blank to call `city_elevation` to obtain the elevation (height above sea level) of Monkton.

```
def city_elevation(city):
    '''Return the elevation of the city (given as a string).'''

    ...  (The rest of the code for this function is not shown.)

    return elevation


print "The elevation of Monkton is", ___________________________________________
```

Use this page for rough work—clearly indicate any section(s) to be marked.

Question 2. [8 MARKS]

Part (a) [4 MARKS]

Complete the function `egg_category` which returns a `str` describing an egg's category given its `int` weight in grams. Here is a table specifying the weight ranges—if an egg's weight is on the boundary between two category ranges, it is assigned the smaller category.

Category	Weight
Small	no more than 50 grams
Medium	50–57 grams

Category	Weight
Large	57–64 grams
Jumbo	more than 64 grams

```
def egg_category(weight):
    '''Return a str describing the category of an egg of the specified int weight.
    '''
```

Part (b) [4 MARKS]

Complete the `main` block below. Your program should use `raw_input` to ask the user for an egg weight and should print the category for that weight, in the form: "`An egg of weight W is a C egg.`", where `W` is the weight the user entered and `C` is the category returned by your function from above. (Note that you can complete this part even if you did not write the function above.)

```
if __name__ == "__main__":
```

Use this page for rough work—clearly indicate any section(s) to be marked.

Question 3. [8 MARKS]

Part (a) [4 MARKS]

Complete the function below according to its docstring.

```
def print_time(sec):
    '''A day has 86400 = 24 * 60 * 60 seconds.  Given an int in the range 0 to 86399,
    print the current time as hours, minutes, seconds on a 24-hour clock.  For example:
    >>> print_time(70000)
    19 h, 26 m, 40 s
    '''
```

Part (b) [4 MARKS]

Fill the table below with four **different** test cases for function `print_time` above—do *not* test for invalid inputs. For each test case, indicate clearly the expected outcome and your reason for choosing this case (in column "Explanation"). Note that you can answer this part even if you did not complete the code above.

Test Case	Expected Outcome	Explanation

Use this page for rough work—clearly indicate any section(s) to be marked.

Question 4. [5 MARKS]

Part (a) [2 MARKS]

Write a suitable docstring for the following function.

```
def func(n):
    '''




    '''
    total = 0
    while n > 0:
        total += n * n
        n -= 1
    return total
```

Part (b) [3 MARKS]

Complete the function below according to its docstring. *(Hint: Look at the string method* `.isdigit()`*.)*

```
def int_input(prompt):
    '''Repeatedly ask the user for a value, using string prompt, until the user enters
    an integer, then return that integer.
    '''
```

Use this page for rough work—clearly indicate any section(s) to be marked.

Question 5. [6 MARKS]

Write a Python program to control the temperature in a building, using the following functions.

- `current_temp()`: Return the current temperature. The initial temperature is `20`, and the temperature remains constant unless it is changed by one of the other functions.
- `raise_temp(deg)`: Raise the temperature by `deg` degrees.
- `lower_temp(deg)`: Lower the temperature by `deg` degrees.

Your solution will be graded on its design as well as its functionality.

Use this page for rough work—clearly indicate any section(s) to be marked.

Question 6. [8 MARKS]

Complete the function below according to its docstring.

```
def longest_sequence(search_str, ch):
    '''Return the length of the longest consecutive sequence of the character ch in the
    string search_str.  For example:
    >>> longest_sequence("aababbbabb", "b")
    3
    >>> longest_sequence("aababbbabb", "a")
    2
    '''
```

On this page, please write nothing except your name.

Family Name(s): _____

Given Name(s): _____