Duration: **110 minutes**
Aids Allowed: **Reference sheet handed out with the test**

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

Family Name(s): _____

Given Name(s): _____

Lecture Section: ☐ LEC01 (Monday, Tuesday, Wednesday)

☐ LEC02 (Monday, Wednesday, Thursday)

---

*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 5 questions and a bonus question 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, 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 to receive full marks, 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.

MARKING GUIDE

# 1:	_____/ 9
# 2:	_____/16
# 3:	_____/ 8
# 4:	_____/ 6
# 5:	_____/ 6
BONUS MARKS:	_____/ 3
TOTAL:	_____/45

Good Luck!

OVER...

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

Question 1. [9 MARKS]

Treat each subquestion independently (*i.e.*, code in one question is not related to code in another), and answer each question in the space provided.

Part (a) Functions [2 MARKS]

Write a function that prints "Happy Halloween" (without the quotes). Also write code that calls the function that you wrote.

Part (b) Loops and Lists [3 MARKS]

Complete the following function. The function should print all the elements of the list `L` that is passed to it as an argument, except the last element. One element should be printed per line. For example, `print_almost_all([41, 42, 43])` should print:

```
41
42
```

```
def print_almost_all(L):
```

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

Part (c) Global Variables [2 MARKS]

Complete the function `h()` so that the effect of running the code below is to print "candy" (without the quotes) and not "pumpkin."

```
def h():



treat = "pumpkin"
h()
print(treat)
```

Part (d) Order of Execution [2 MARKS]

What is the output of the following piece of code?

```
x = 4
y = x + 5
x = 7
print(x, y)
```

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

Question 2. [16 MARKS]

Part (a) [4 MARKS]

Complete the following function. The function returns the number of kids whose favourite part of Halloween is candy, according to the data in the dictionary `faves`. For a string `name`, `faves[name]` contains the favourite part of Halloween of the kid named `name`. Both the names and the favourites are lowercase strings. For example,

`count_candy({"ben": "costumes", "adam":"candy", "matt":"candy", "anna":"costumes"})`
returns 2, since two kids (adam and matt) like candy. Assume `faves` is given in the correct format.

```
def count_candy(faves):
```

Part (b) [4 MARKS]

Write code that prints all the integers from -500 to 500, from smallest to largest. That is, the code should print all of

$-500, -499, -498, ..., -2, -1, 0, 1, 2, 3, ..., 499, 500.$

Each integer should be printed on a separate line.

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

Part (c) [4 MARKS]

Complete the following function according to its docstring.

```
def get_key_by_val(d, val):
  """Return a key in the dictionary d such that d[key] == val. If there are several
  such keys, return one of them (it doesn't matter which). If there are no such
  keys, return None."""
```

Part (d) [4 MARKS]

Complete the following function according to its docstring.

```
def all_not_same(a, b, c):
  """Return True if a, b, and c are **not** all equal to each other. Return
  False otherwise."""
```

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

Question 3. [8 MARKS]

Each of these subquestions contains a piece of code. Treat each piece of code independently (*i.e.*, code in one question is not related to code in another), and **write the expected output for each piece of code**. If the code produces an error, write down the output that the code prints before the error is encountered, and then write "ERROR." You do not have to specify what kind of error it is.

Part (a) [2 MARKS]

```
def f(n):
  print(n)

m = 2
f(m)
```

Part (b) [2 MARKS]

```
def f(n):
  n = 5

n = 2
f(n)
print(n)
```

Part (c) [2 MARKS]

```
def f(L):
  L[0] = 42
  print(L[0])

L = [1, 2, 3]
f(L)
print(L[0])
```

Part (d) [2 MARKS]

```
d1 = {1:[2], 3:[4]}
d2 = d1
d1 = {1: d2[1], 3: d2[3]}
d1[1][0] = 5
print(d1[1][0], d2[1][0])
```

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

Question 4. [6 MARKS]

Complete the function below according to its docstring. For full marks, do **not** use `list`'s method `.extend()` anywhere in your solution. Solutions that use `.extend()` will get at most 3 marks.

```
def flatten(list_of_lists):
    """Return a new list that contains every value in each of the sub-lists of
    list_of_lists.  For example,
    >>> flatten([[1, 3], ['a', 'b', 'c']])
    [1, 3, 'a', 'b', 'c']
    Assume that all the elements of list_of_lists are lists and that the elements
    of the elements of list_of_lists are integers or strings."""
```

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

Question 5. [6 MARKS]

Write code that prints out all the possible 4-letter strings that can be written using the letters "a", "b", "c", "d", "e", "f" and "g" (and no other character) such that no letter appears in any string more than twice. For example, `"ccbb"` should be printed, but `"aaab"` should not. Hint: think about how you would print *all* the 4-letter strings.

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

Bonus. [3 MARKS]

Write a function with the signature `near_anagram(w1, w2)` that takes in two strings, `w1` and `w2`, and returns `True` iff `w1` can be obtained from `w2` by rearranging the characters in `w2`, and then changing exactly one of the characters for another character. For example `near_anagram("cat", "tap")` is `True` since `"cat"` can be rearranged into `"tac"`, and then changed into `"tap"` by exchanging `c` for `p`. Assume `w1` and `w2` only contain lowercase letters.

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

On this page, please write nothing except your name.

Family Name(s): _____

Given Name(s): _____