Duration: **120 minutes**
Aids Allowed: **Reference sheet handed out with the test**

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

Family Name(s): _____

Given Name(s): _____

UofT email: _____@mail.utoronto.ca

Lecture Section: ☐ LEC0101 (M5, W2, R3)

☐ LEC0102 (M4, W4, R11)

---

*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 7 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, 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 *might* be given for partial solutions with comments clearly indicating what the missing parts should accomplish.

You may use functions written to answer one question as part of the solution to another question.

You may **not** `import` any modules unless otherwise specified.

MARKING GUIDE

1: _____/ 8

2: _____/ 9

3: _____/ 8

4: _____/ 4

5: _____/ 8

6: _____/ 4

7: _____/ 4

TOTAL: _____/45

Good Luck!

OVER...

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

Question 1. [8 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) [2 MARKS]

Complete the following function. The function takes in a string `thing`, and *returns* the string `"NOO!"` if `thing` is `"ghost"`, `"monster"`, or `"midterm"`, and the string `"YAY!"` otherwise. For example, the call `halloween_reaction("ghost")` should return the string `"NOO!"`, and the call `halloween_reaction("candy")` should return the string `"YAY!"`.

```
def halloween_reaction(thing):
```

Part (b) [2 MARKS]

Complete the following function. The function takes in a list `L`, and prints all the elements of the list `L`, in order, except for the first element and the last element. One element should be printed per line. For example,

```
print_mid_part(["pumpkins", "candy", "costumes", "autumn", "zombies"])
```

should print

```
candy
costumes
autumn
```

```
def print_mid_part(L):
```

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

Part (c) [2 MARKS]

Complete the function `h()` so that the effect of running the code below is to print
`["fall", "colours"]`
(and nothing else).

```
def h(L):
```

```
if __name__ == "__main__":
    L = ["tricks", "treats"]
    h(L)
    print(L)  #should print ["fall", "colours"]
```

Part (d) [2 MARKS]

What is the output of the following piece of code?

```
x = "Spice"
y = "Pumpkin"
x += x
print(y+x, "Latte")
```

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

Question 2. [9 MARKS]

Part (a) [5 MARKS]

Complete the following function, which computes the sum of all the elements in the list `L` that are odd (i.e., not divisible by 2.) For example, if `L` is `[1, 3, 4, 5]`, the function should return 9 since $9 = 1 + 3 + 5$. Assume `L` is a list of integers.

```
def odds_sum(L):
   """Return the sum of the odd elements of L"""
```

Part (b) [4 MARKS]

The following code is written using a `for`-loop. Write it using a `while`-loop instead.

```
for i in range(5, 500, 3):
  print(i)
```

A version of the code above written using a `while`-loop:

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

Question 3. [8 MARKS]

Part (a) [4 MARKS]

Complete the following function. The function takes in a list of strings `faves`, and a list of string, of the same length, `kids`. The favourite thing about Halloween of the kid whose name is `kids[i]` is `faves[i]`. The function returns the list of the names of the kids whose favourite thing about Halloween is `"candy"` (`"candy"` is in lowercase). The names in the list that the function returns should appear in the same order that the names appear in the list `kids`. For example, if

```
faves ==  ["candy", "costumes", "weather", "candy"] and
kids  ==  ["Bob",   "Dorothy",  "Mike",    "Alice"],
```

then `kids_who_like_candy(faves, kids)` should return the list `["Bob", "Alice"]`.

```
def kids_who_like_candy(faves, kids):
```

Part (b) [4 MARKS]

Complete the following function. The function returns the cube root of n (i.e., $\sqrt[3]{n}$.) **Assume that $\sqrt[3]{n}$ is an integer.** You may **not** `import` any modules or use the `**` operator.

```
def cube_root(n):
```

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

Question 4. [4 MARKS]

Complete the following code in such a way that the output is as stated in the comments.

```
def halloween_surprise():
```

```
if __name__ == '__main__':
  print(halloween_surprise()) #Output: 3
  print(halloween_surprise()) #Output: 2
  print(halloween_surprise()) #Output: 1
  print(halloween_surprise()) #Output: SURPRISE!
```

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

Question 5. [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]

```
L1 = [1, 2]
L2 = L1[:]
L2 = [3, 4]
print(L1)
```

Output:

Part (b) [2 MARKS]

```
def f():
  n = 5

n = 4
n = f()
print(n)
```

Output:

Part (c) [2 MARKS]

```
def f(L):
  L2 = L
  L = [1, 2]
  L[0] = 5
  print(L)

L = [2, 3]
f(L)
print(L[0])
print(L2)
```

Output:

Part (d) [2 MARKS]

```
L1 = [[[1, 2], 3], 4]
L2 = L1[:]
L1[1] = 5
L1[0][1] = 5
L1[0][0][1] = 5
print(L2)
```

Output:

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

Question 6. [4 MARKS]

Write a function that determines if a list has only a single "peak." A list has a single peak if the list is non-decreasing up to a certain point, and is non-increasing after that point. For example, the list `[1, 2, 2, 3, 4, 5, 0, -1]` has a single "peak," since it is non-decreasing up to `5`, and non-increasing after `5`. On the other hand, the list `[1, 2, 1, 2]` has more than one "peak," so we say that it is not true that it has only a single "peak." Non-decreasing lists like `[1, 2, 3]` and non-increasing lists like `[3, 2, 1]` are considered to have a single "peak."

```
def has_single_peak(L):
  """Return True iff the list of integers L has only a single peak"""
```

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

Question 7. [4 MARKS]

Write a function with the signature `def max_arrivals_2hrs(arrivals)` that returns the maximum number of arrivals of trick-or-treating kids that happened within the span of two hours, as recorded in the list `arrivals`. The list `arrivals` is a list of times (in minutes) after 5PM on Oct. 31 that kids arrived trick-or-treating. For example, if `arrivals` is `[0, 30, 40, 150, 160, 170, 370]`, then kids arrived at 5:00PM, 5:30PM, 5:40PM, 7:30PM, 7:40PM, etc.; and the maximum number of arrivals within the span of two hours was 3 (the arrivals at 5:00PM, 5:30PM, and 5:40PM.) You can assume that the list `arrivals` contains only integers and is non-decreasing.

```
def max_arrivals_2hrs(arrivals):
```

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): _____