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

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

UTORid: _____

Family Name(s): _____

Given Name(s): _____

UofT email: _____@mail.utoronto.ca

Lecture Section: ☐ LEC0101 (M11, T5, F12)

☐ LEC0102 (M12, T12, F2)

---

*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 on 8 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. *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: _____/20

2: _____/10

3: _____/10

4: _____/10

5: _____/12

TOTAL: _____/62

Good Luck!

OVER...

Question 1. [20 MARKS]

Part (a) [10 MARKS]

Write code to print all the even numbers between 100 and 200 (including 200). Your code should print one number per line.

Part (b) [10 MARKS]

Complete the following function.

```
def last_ind(L, e):
   '''Return the largest index at which element e appears in the list L.
   If e is not in L, return None.
   >>> last_ind([1, 5, 3, 5, 4], 5)
   3

   >>> last_ind([1, 2, 3], 0)
   None
   '''
```

Question 2. [10 MARKS]

Recall that a sparse matrix can be stored as a dictionary. For example, the matrix

$$\begin{pmatrix} 5 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

can be stored as the dictionary `M = {(0, 0): 5, (1, 2): 1}`. We can store the dimension of a matrix in a tuple. For example, the dimension of `M` can be stored as `(2, 3)`. Recall that a matrix can also be stored as a list of lists, with each list representing a row. For example, the matrix above can be stored as `[[5, 0, 0], [0, 0, 1]]`.

Complete the following function.

```
def add_sparse_matrices(A, B, dim):
    '''Return the sum of the two sparse matrices A and B, which are of dimension dim.
      A and B are stored as dictionaries, and dim is a tuple with two elements.
      Return the result as a list of lists rather than as a dictionary'''
```

Question 3. [10 MARKS]

Write code to find the smallest common multiple of four input integers. The smallest common multiple of the positive integers a, b, c, d is the smallest positive integer that is divisible by all of a, b, c, and d.

CONT'D...

Question 4. [10 MARKS]

Write a function that can be called repeatedly with a string argument. The function should return `"locked"`, unless it is being called with the argument `"pumpkin"`, and it was called with the arguments `"fall"`, `"costumes"`, `"costumes"`, in that order, immediately before, in which case the function should return `"unlocked"`.

For example,

```
unlock("ghosts")     # returns "locked"
unlock("leaves")     # returns "locked"
unlock("fall")       # returns "locked"
unlock("costumes")   # returns "locked"
unlock("costumes")   # returns "locked"
unlock("pumpkin")    # returns "unlocked"
unlock("pumpkin")    # returns "locked"
unlock("fall")       # returns "locked"
unlock("costumes")   # returns "locked"
unlock("costumes")   # returns "locked"
unlock("pumpkin")    # returns "unlocked"
unlock("fall")       # returns "locked"
```

```
def unlock(input):
```

Question 5. [12 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. For each subquestion, you will earn either 3 or 0 marks.

Part (a) [3 MARKS]

```
def f(L):
  L = [1, 2, 3]
  L[0] = 5
  print(L[0])

L1 = [4, 5, 6]
f(L1)
print(L[0])
```

Output:

Part (b) [3 MARKS]

```
def g(n):
  n = 3
  print(n)

m = 5
m = g(m)
print(m)
```

Output:

Part (c) [3 MARKS]

```
def h(L):
  L[1] = [7, 8]
  L[0] = [5, 6]
  L[0][0] = 9
  print(L)


L = [[1, 2], [3, 4]]
h(L)
print(L)
```

Output:

Part (d) [3 MARKS]

```
L = [[[1, 2], [3, 4]]]
L1 = L[:]
L1[0][0] = 5
L1[0][1][0] = 6
print(L1)
print(L)
```

Output:

On this page, please write nothing except your name.

Family Name(s): _____

Given Name(s): _____