Name _____ Student No. _____

No aids allowed. Answer all questions on test paper. Use backs of sheets if necessary.

Total Marks: **50**

[10] 1. Answer *briefly* the following questions:

(a) What is the Python *Global Interpreter Lock* (GIL)? Make sure you mention "ticks."

(b) How does the GIL relate to threads?

(c) What Python solution would you propose for *I/O-bound* tasks versus *CPU-bound* tasks?

(d) What is the difference between the Python *Lock* synchronization mechanism, and the Python *RLock* (*Re-entrant Lock*) sync. mechanism?

2. Consider the code given in `ex2.py` on the next page. What is the difference in output between the following two commands:

   (a) `python ex2.py T`

   (b) `python ex2.py F`

   Explain your answer.

ex2.py

```
# ex2.py
# Daemon threads
# Feb 12, 2013
# SE3BB4
# Michael Soltys

import sys, threading, time

class DaemonThread(threading.Thread):
    def __init__(self, turn_on_daemon):
        threading.Thread.__init__(self)
        self.daemon = turn_on_daemon

    def run(self):
        x = 0
        while 1:
            if self.daemon:
                print "Daemon step: %d" % x
            else:
                print "Non-Daemon step: %d" % x
            x += 1
            time.sleep(1)

if __name__ == "__main__":
    print "__main__ start"
    if sys.argv[1] == "T":
        thread = DaemonThread(True)
    else:
        thread = DaemonThread(False)
    thread.start()
    time.sleep(5)
    print "__main__ stop"
```

3. Consider the code given in `ex3.py` on the next page. What is the difference in output between the following two commands:

   (a) `python ex3.py`

   (b) `python ex3.py the sound of music`

   Explain your answer.

ex3.py

```
1  # ex3.py
2  # An example of join
3  # Feb 12, 2013
4  # SE3BB4
5  # Michael Soltys
6
7  import sys, threading, datetime, time
8
9  class DateThread(threading.Thread):
10
11   def __init__(self, wait):
12     threading.Thread.__init__(self)
13     self.wait = wait
14
15   def run(self):
16     now = datetime.datetime.now()
17     print "%s says Hello World at time: %s" % (self.getName(), now)
18     time.sleep(self.wait)
19     print "%s slept for %d seconds" % (self.getName(), self.wait)
20
21 for i in range(5):
22   t = DateThread(i+1)
23   t.start()
24   if (len(sys.argv)>1):
25     t.join()
```

4. Consider the code given in `ex5.py` on the next page. Explain the effect of the *Lock* mechanism. Also, list the possible ways in which the file `output.html` might be written into — mention the concept of *shuffling* (a.k.a., *interleaving*).

ex5.py

```
1  # ex5.py
2  # Fetching URLs with locks
3  # Feb 22, 2013
4  # SE3BB4
5  # Michael Soltys
6
7  import time, urllib2, threading
8
9  class GetURLs(threading.Thread):
10
11   def __init__(self, urls, output, lock):
12     threading.Thread.__init__(self)
13     self.urls = urls
14     self.output = output
15     self.lock = lock
16
17   def run(self):
18     while self.urls:
19       url = self.urls.pop()
20       req = urllib2.Request(url)
21       try:
22         d = urllib2.urlopen(req)
23       except urllib2.URLError, e:
24         print 'URL %s failed: %s' % (url, e.reason)
25       self.lock.acquire()
26       print 'lock acquired by %s' % self.name
27       self.output.write(d.read())
28       print 'write finished by %s' % self.name
29       print 'lock released by %s' % self.name
30       self.lock.release()
31       print 'URL %s fetched by %s' % (url, self.name)
32
33 def main():
34   urls1 = ['http://www.math.mcmaster.ca']
35   urls1[len(urls1):] = ['http://www.cas.mcmaster.ca']
36   urls2 = ['http://dailynews.mcmaster.ca']
37   urls2[len(urls2):] = ['http://www.mcmaster.ca']
38   lock = threading.Lock()
39   f = open('output.html', 'w+')
40   t1 = GetURLs(urls1, f, lock)
41   t2 = GetURLs(urls2, f, lock)
42   t1.start()
43   t2.start()
44   t1.join()
45   t2.join()
46   f.close()
```

[10] 5. Give Petri-Nets models for the following two systems:

(a) A single elevator in a 5 floor building. You only have to describe the following: the elevator goes up or down, by a single floor; it cannot move up on the 5th floor, and it cannot go down on the 1st floor.

(b) An ATM where you insert a debit card, and provide a PIN of 4 numbers, and press 'OK'. If the PIN is incorrect, of 'OK' is pressed out of sequence, or too many numbers are entered, the card should be rejected. Otherwise, accepted.