1. (30 points) **XML Data** Consider the following data in the form of a tree.

```
X
A
  A
    B C
      one four
    two three
  B
    B
    five six
A
  A
    B
    seven
```

(a) (12 points) Write down the XML document that describes this tree.

(b) (18 points) For each of the XPath expressions below indicate what they return when evaluated on the data represented above. For example, if the expression is: `/X/A/B/B/text()` then you would answer: two, three

i. `/X/A/C/text()`
ii. `/X/A/*/text()`
iii. `//A/B/*/text()`
iv. `/X/A[A]/B/*/text()`
v. `//A[B/A]/C/text()`
vi. `/X/A[A][C]/B/*/text()`
2. (70 points) **Buffer Management**


The provided code includes implementations of three replacement policies: RANDOM, LRU (Least Recently Used), and MYSTERY. The code simulates the operation of the buffer manager in response to a workload of page requests. At the end of the simulation, you can compute the fraction of cache misses. You will use the code to analyze the fraction of cache misses for different policies under two different workloads of page requests and for varying buffer sizes.

Please carry out steps (a) and (b), and then answer the remaining questions:

(a) Modify the Python code to implement two new policies: MRU (Most Recently Used) and FIFO (First-In First-Out). Note that the MRU and LRU policies make their choice by the time of last ‘use’ of a page in the buffer. The FIFO policy depends on the time the page entered the buffer. You do not need to worry about the efficiency of your implementation. Stub function calls are provided in the code. (You will turn in this code.)

(b) Simulate both the random and nested-loops workload for each policy by running the modified code. Use the output to complete the tables in the provided Excel spreadsheet. (You will turn in the final spreadsheet.)

(c) Why does the cache miss rate remain above zero, even when the buffer is as large as the total number of data pages (i.e. the value of dataSize)?

(d) How does policyMYSTERY compute the page to replace and how does it perform in comparison to the other policies? What challenge would you face in implementing policyMYSTERY in a real system?

(e) For the nested loops workload, how would you explain the performance of the MRU policy when the buffer size is greater than or equal to 50?

**Turn-in for this assignment**

Please submit through SPARK the following three files:

1. A document (e.g. text, PDF, etc.) containing your written answers to Problem 1 and Problem 2 (c), (d), and (e).

2. Python code, as described in Problem 2(a). **The main function in your code should be set to run all policies under the nested loops workload.**

3. Excel spreadsheet with values filled in following Problem 2(b).