University of Toronto Mississauga

Sample Synchronization Midterm
Course: CSC369H5F
Instructor: Andrew Petersen
Duration: 45 minutes
Aids allowed: None

Last Name: _____

Given Name: _____

Student Number: _____

This midterm test consists of 2 questions on 6 pages (including this one). Please write legibly and be as specific as possible. Precise answers will be given higher marks than vague ones, and marks will be deducted for any incorrect statements in an answer.

If you need extra space or scratch paper, raise your hand, and the instructor will bring you a few sheets.

Marking Guide

1: _____/10

2: _____/ 6

TOTAL: _____/16

continued on page 2

10 marks **Question 1.** Short Answer

2 marks **Part (a)**

Define the term "mutual exclusion".

3 marks **Part (b)**

Draw a diagram of the address space of a program with two threads. Label all of the major sections and indicate where the PC and SP registers point.

CONTINUED ON PAGE 3

2 marks

Part (c)

Discuss the trade-off between having many locks protecting small pieces of data and a single lock protecting all critical sections. (For example, in assignment 1, you could have added a lock to each file block in the cache, or you could have created a single lock for the entire file table.)

2 marks

Part (d)

Add appropriate synchronization to the following threads so that `resource--` is executed if and only if `resource` is greater than 0. Your solution should not enforce any other constraints. Be sure to declare and initialize any synchronization variables.

```
// Synchronization variables and initialization
```

```
// Thread X                          // Thread Y

resource--;                          resource++;
```

CONTINUED ON PAGE 4

6 marks

Question 2. Synchronization Problem

Hungry, hungry hippos! There are `H` hippos (4, in the board game, but we can imagine more). A referee dumps a handful of marbles (`M` of them) between the hippos, and the hippos attempt to consume marbles as quickly as possible. However, each hippo can only consume one marble at a time. When all of the marbles have been consumed, the hippo who ate the last marble informs the referee, who prints the score for that round and then dumps `M` more marbles onto the board. This game repeats forever.

The basic algorithm for the game is given below, but it lacks any synchronization. Fill in the appropriate synchronization and any conditionals (if-statements) required to protect the global data. You will need to declare your synchronization primitives, but you may assume they are correctly initialized elsewhere.

Hint: Use a monitor implemented with condition variables (`CV_wait` and `CV_signal`). Assume these condition variables follow the *Mesa* convention.

```
hippo(int ID) {
    while (1) eat(ID);
}

referee() {
    while (1) restart_game();
}

// GLOBALS
int marbles = M;
int scores = {0, 0, 0, ... 0};

// SYNCHRONIZATION DECLARATIONS
```

CONTINUED ON PAGE 5

```
void eat(int ID) {









    marbles--;
    scores[ID]++;









}

void restart_game() {









    print_and_clear_scores();
    marbles = M;









}
```

CONTINUED ON PAGE 6

*[Use the space below for rough work. This page will **not** be marked, unless you clearly indicate the part of your work that you want us to mark.]*

END OF EXAMINATION