CPSC 561, Fall 2017 16. November 2017

University of Calgary

Homework Assignment # 2

Due: 27. November 2017, 11:59am

The Rules: You are allowed to discuss the problems with other students currently enrolled in CPSC 561, or with your TA or instructor. But you are not allowed to use the help of any other person. If you discuss possible solutions with other students enrolled in this course, you must list the names of those students on the cover sheet (see the last page of this assignment). You are also **not allowed to take away any notes made during these discussions**. Instead, you have to write up the solutions entirely on your own.

You can use published literature, as long as you cite that literature in a scientific way, including page numbers, URLs, etc. (whether you use published literature or not will not affect your grade). However, in general it should not be necessary to consult literature other than the course textbook or the lecture notes. In any case, **your solutions must be self-contained**, meaning that a person with knowledge of the course material must be able to follow all your reasoning without referring to any external literature. For example, if you use a (not well-known) statement, say from Wikipedia, which was not covered in the lecture, then you have to provide a proof of that statement as part of your solution.

If you are unsure whether a certain source can be used, **please ask your instructor**.

Please fill out the cover sheet (see the last page) and hand it in with your solutions.

Question 1
An LL/SC object stores arbitrary values and provides two atomic operations, `LL()` and `SC()`. An `LL()` operation returns the value of the object. An `SC(`x`)` operation takes a parameter, x, and may be successful and return `True` or unsuccessful and return `False`. If successful, the `SC(`x`)` operation changes the value of the object to x, and if unsuccessful it does not affect the value of the object. An `SC()` operation by process p is successful if and only if p has previously called `LL()` and no successful `SC()` operation has occurred since then.

A CAS object provides a `Comp&Set()` operation as defined in the textbook on p. 113 and an operation `Read()`, which returns the current value of the object. An SCAS object also provides a `Comp&Set()` and a `Read()` operation, and it is only different from a CAS object in that the `Comp&Set()` operation always returns the previous value of the object (instead of `True` or `False`).

(a) Implement a linearizable CAS object from atomic SCAS objects and registers.

(b) Implement a linearizable SCAS object from atomic CAS objects and registers.

(c) Implement a linearizable CAS object from atomic LL/SC objects and registers.

Your implementations have to be wait-free, and you have to prove that they are wait-free and linearizable. You must give direct implementations, and not use other algorithms, such as the universal construction. Try to find implementations, where each operation takes only a constant number of steps. (You can still get partial marks, if your implementations are less efficient than that.)

Question 2
A $\mathbb{Z}_k$-counter stores an integer in $\{0, \ldots, k-1\}$ and supports one operation, `Inc()`. If the value of the object is $x \in \{0, \ldots, k-1\}$, then `Inc()` changes the value of the object to $(x+1) \bmod k$ and returns x.

Show for any $k \geq 2$ that $\mathbb{Z}_k$-counters have consensus number 2.

Class `SRSW_Register`

shared: $B[0 \dots 7] := (0,0,0,0,1,0,0,0)$ of type `SRSW_Bit`

Method `Write`(x)

$B[x]$.`Write(1)`
for $i = x-1, \dots, 0$ **do**
  $B[i]$.`Write(0)`
end

Method `Read()`

$i = 0$
while $B[i]$.`Read()`$=0$ **do**
  $i := i+1$
end
$val := i$
for $i = val - 1, \dots, 0$ **do**
  if $B[i]$.`Read()` $= 1$ **then** $val := i$
end
return val

Figure 1: Implementation of a linearizable SRSW 8-valued register from atomic bits

Question 3

Let R be an SRSW register that stores values in $\{0, \dots, 7\}$ and is initialized to 4. Consider two processes, w and r, executing the following (randomized) algorithm:

w: R.`Write`(5); choose $c \in \{1, 6\}$ uniformly at random; R.`Write`(c);

r: R.`Read()`

(a) Suppose R is atomic. Describe an adaptive adversary that schedules w and r in such a way that the expectation of the value read by r is as small as possible. What is that expected value?

(b) Now suppose that R is implemented from an array B of atomic Boolean registers as depicted in Figure 1.

For this implementation of R, describe an adaptive adversary that schedules w and r in such a way that the expectation of the value read by r is as small as possible. What is that expected value?

Cover Page for CPSC 561 Homework Assignment # 2

Name: ______________________________

Course Section: ______________________________

Collaborators:

Question 1: ______________________________

Question 2: ______________________________

Question 3: ______________________________

Other Sources:

Question 1: ______________________________

Question 2: ______________________________

Question 3: ______________________________

Declaration:

I have written this assignment myself. I have not copied or used the notes of any other student.

Date/Signature: ______________________________