First Name: ________________________ Last Name: ________________________

Sample Test #1 — CSc 422, Fall 2017

This is a 50 minute exam, and contains 6 pages and four questions worth 15 points each

Answer **THREE** of the four questions.

Either leave one question blank, or mark one question Do Not Grade.

It is a closed-book exam. Calculators and foreign-language-to-English dictionaries are allowed. Put your answers in the space provided here. Show your work.

You may use one sheet of paper, each no larger than 8.5” x 11”, with notes written in your own handwriting. The notes may be on both sides of the paper. Turn in the notes page with your test.

You *must* explain your answers or show how you arrived at them. This is required for full credit and is helpful for partial credit.

1. (15 points) — Consider the following variation on the producer-consumer problem: Assume one producer process and **N** consumer processes share a single buffer that can hold one item. The producer deposits an item into the buffer. Each of the consumers then fetches a copy of the item. Each item deposited by the producer has to be fetched (copied) by all **N** consumers before the producer can deposit another item into the buffer. Develop a solution for this variation of the producer-consumer problem using <u>semaphores</u> for synchronization.

2. (15 points) — The first printing of the textbook has the following for the two-process, tie-breaker algorithm (also known as “Peterson’s Algorithm”):

```
bool in1 = false, in2 = false;
int  last = 1;
```

`process CS1 {`
`   while (true) {`
`      last = 1;`
`      in1 = true;`
`      while (in2 and last == 1)`
`         skip;`
`      critical section`
`      in1 = false;`
`      non-critical section`
`   } # while`
`} # CS1`
	`process CS2 {`
	`   while (true) {`
	`      last = 2;`
	`      in2 = true;`
	`      while (in1 and last == 2)`
	`         skip;`
	`      critical section`
	`      in2 = false;`
	`      non-critical section`
	`   } # while`
	`} # CS2`

The difference between the above printing and the correct solution is in the order of the assignments to **`last`** and **`in1`** and **`in2`**. The correct solution has the assignment to **`last`** <u>after</u> the assignment to **`in1`** or **`in2`**.

<u>Explain</u> why the incorrect solution is incorrect.

3. (15 points) — Consider the following three statements:

```
S1:  x = x + y;
S2:  y = x - y;
S3:  x = x - y;
```

Assume that **x** is initially **2** and that **y** is initially **5**. For each of the following, what are the possible final values of **x** and **y**? Explain your answers.

a.)

```
S1; S2; S3;
```

b.)

```
co < S1; >
// < S2; >
// < S3; >
oc
```

c.)

```
co < await (x > y) S1; S3; >
// < S2; >
oc
```

4. (15 points) — There are two parts to this question. Part b.) is on the next page.

Consider the following Readers/Writers solution (taken from the lecture slides). The solution is a "Reader's Preference" solution.

a.) **(5 points)** Modify the solution to make it a "Writer's Preference" solution.

```
int nr = 0, nw = 0
sem e = 1, r = 0, w = 0;
int dr = 0, dw = 0;
```

```
process Reader[i = 1 to M] {
   while ( true ) {
      # <await (nw == 0)
           nr = nr + 1; >
      P(e);  # Pick up baton
      if ( nw > 0 ) {
         dr = dr + 1;
         V(e); P(r);
      }
      nr = nr + 1;
      if ( dr > 0 ) {
         dr = dr - 1;
         V(r);
      }
      else
         V(e);
      read the database;
      # < nr = nr - 1; >
      P(e);  # Pick up baton
      nr = nr - 1;
      if ( nr == 0 and dw > 0 ) {
         dw = dw - 1;
         V(w);
      }
      else
         V(e);
   } # while
} # Reader
```

```
process Writer[j = 1 to N] {
   while ( true ) {
      # < await (nr == 0 and
                 nw == 0)
             nw = nw + 1; >
      P(e);  # Pick up baton
      if ( nr > 0 or nw > 0 ) {
         dw = dw + 1;
         V(e); P(w);
      }
      nw = nw + 1;
      V(e);
      write the database;
      # < nw = nw - 1; >
      P(e);  # Pick up baton
      nw = nw - 1;
      if ( dr > 0 ) {
         dr = dr - 1;
         V(r);
      }
      elseif ( dw > 0 ) {
         dw = dw - 1;
         V(w);
      }
      else
         V(e);
   }
}
```

b.) **(10 points)** This is the same “Reader’s Preference” solution.

Modify this to alternate between Reader’s and Writer’s as follows: Reader’s have preference until 5 readers have entered the database. Once 5 readers have entered, preference is given to writers until one writer has entered the database.

Note: if there are only readers and no writers, all readers should get in; and if there are only writers and no readers, all writers should get in.

```
int nr = 0, nw = 0
sem e = 1, r = 0, w = 0;
int dr = 0, dw = 0;
```

```
process Reader[i = 1 to M] {
   while ( true ) {
      # <await (nw == 0)
           nr = nr + 1; >
      P(e);  # Pick up baton
      if ( nw > 0 ) {
         dr = dr + 1;
         V(e); P(r);
      }
      nr = nr + 1;
      if ( dr > 0 ) {
         dr = dr - 1;
         V(r);
      }
      else
         V(e);
      read the database;
      # < nr = nr - 1; >
      P(e);  # Pick up baton
      nr = nr - 1;
      if ( nr == 0 and dw > 0 ) {
         dw = dw - 1;
         V(w);
      }
      else
         V(e);
   } # while
} # Reader
```

```
process Writer[j = 1 to N] {
   while ( true ) {
      # < await (nr == 0 and
                 nw == 0)
            nw = nw + 1; >
      P(e);  # Pick up baton
      if ( nr > 0 or nw > 0 ) {
         dw = dw + 1;
         V(e); P(w);
      }
      nw = nw + 1;
      V(e);
      write the database;
      # < nw = nw - 1; >
      P(e);  # Pick up baton
      nw = nw - 1;
      if ( dr > 0 ) {
         dr = dr - 1;
         V(r);
      }
      elseif ( dw > 0 ) {
         dw = dw - 1;
         V(w);
      }
      else
         V(e);
   }
}
```

Solutions to Sample Test #1 — CSc 422, Fall 2017

1. (15 points) — Consider the following variation on the producer-consumer problem: Assume one producer process and **N** consumer processes share a single buffer that can hold one item. The producer deposits an item into the buffer. Each of the consumers then fetches a copy of the item. Each item deposited by the producer has to be fetched (copied) by all **N** consumers before the producer can deposit another item into the buffer. Develop a solution for this variation of the producer-consumer problem using semaphores for synchronization.

Solution:

```
sem full[N] = ([N] 0)  # create N semaphores, set each to 1
sem empty = N;  # only need one of these, set to N
```

Producer:

```
   while (true) {
      Produce an item
      for (i = 1 to n)
         P(empty);
      put item in buffer
      for (i = 1 to n)
         V(full[i]);
   } # while
```

Consumer[id = 1 to n]:

```
   while (true) {
      P(full[id]);
      get item from buffer
      V(empty);
      Consume item
   }
```

2. (15 points) — The first printing of the textbook has the following for the two-process, tie-breaker algorithm (also known as "Peterson's Algorithm"):

```
bool in1 = false, in2 = false;
int  last = 1;
```

process CS1	process CS2

`process CS1 {`
`   while (true) {`
`      last = 1;`
`      in1 = true;`
`      while (in2 and last == 1)`
`         skip;`
`      critical section`
`      in1 = false;`
`      non-critical section`
`   } # while`
`} # CS1`
	`process CS2 {`
	`   while (true) {`
	`      last = 2;`
	`      in2 = true;`
	`      while (in1 and last == 2)`
	`         skip;`
	`      critical section`
	`      in2 = false;`
	`      non-critical section`
	`   } # while`
	`} # CS2`

The difference between the above printing and the correct solution is in the order of the assignments to **`last`** and **`in1`** and **`in2`**. The correct solution has the assignment to **`last`** after the assignment to **`in1`** or **`in2`**.

Explain why the incorrect solution is incorrect.

Solution:

To show the solution is incorrect, it is sufficient to show (at least) one situation where the solution fails.

CS1 executes **`last = 1`**

CS2 executes **`last = 2`**

CS2 executes **`in2 = true`**

CS2 executes **`while (in1 and last == 2)`**. The condition is false, since **`in1`** is false. The **`while`** loop exits and CS2 enters the critical section.

CS1 executes **`in1 = true`**

CS1 executes **`while (in2 and last == 1)`**. The condition is false, since **`last`** is 2. The **`while`** loop exits and CS1 enters the critical section.

Both processes are now in the critical section.

3. (15 points) — Consider the following three statements:

```
S1:  x = x + y;
S2:  y = x - y;
S3:  x = x - y;
```

Assume that **x** is initially **2** and that **y** is initially **5**. For each of the following, what are the possible final values of **x** and **y**? Explain your answers.

a.)

```
S1; S2; S3;
```

b.)

```
co < S1; >
// < S2; >
// < S3; >
oc
```

c.)

```
co < await (x > y) S1; S3; >
// < S2; >
oc
```

Solution:

a.) **x = 2 + 5 = 7; y = 7 - 5 = 2; x = 7 - 2 = 5**. Thus, **x = 5**, **y = 2**.

b.) There are 6 possible sequences to consider:

S1; S2; S3: this is the answer from part a. **x = 5**, **y = 2**.
S1; S3; S2: x = 2 + 5 = 7; x = 7 - 5 = 2; y = 2 - 5 = -3. Thus, **x = 2**, **y = -3**.
S2; S1; S3: y = 2 - 5 = -3; x = 2 + -3 = -1; x = -1 - -3 = 2. Thus, **x = 2**, **y = -3**.
S2; S3; S1: y = 2 - 5 = -3; x = 2 - -3 = 5; x = 5 + -3 = 2. Thus, **x = 2**, **y = -3**.
S3; S1; S2: x = 2 - 5 = -3; x = -3 + 5 = 2; y = 2 - 5 = -3. Thus, **x = 2**, **y = -3**.
S3; S2; S1: x = 2 - 5 = -3; y = -3 - 5 = -8; x = -3 + -8 = -11. Thus, **x = -11**, **y = -8**.

c.) Initially, the first arm of the **co** is blocked, since 2 > 5 is false. The second arm of the **co** executes and computes: **y** = 2 - 5 = **-3**. The first arm of the **co** will now find **x > y** is true, since 2 > -3 is true. S1 and S3 are executed in that order. S1 sets **x** = 2 + -3 = **-1**. S3 sets **x** = -1 - -3 = **+2**. Thus, the final values are:

```
x = 2
y = -3
```

4. (15 points) — There are two parts to this question. Part b.) is on the back of this page. Consider the following Readers/Writers solution (taken from the lecture slides). The solution is a "Reader's Preference" solution.

a.) **(5 points)** Modify the solution to make it a "Writer's Preference" solution.

b.) **(10 points)** This is the same "Reader's Preference" solution. Modify this to alternate between Reader's and Writer's as follows: Reader's have preference until 5 readers have entered the database. Once 5 readers have entered, preference is given to writers until one writer has entered the database. Note: if there are only readers and no writers, all readers should get in; and if there are only writers and no readers, all writers should get in.

Solution:

a.)

```
int nr = 0, nw = 0
sem e = 1, r = 0, w = 0;
int dr = 0, dw = 0;
```

```
process Reader[i = 1 to M] {
   while ( true ) {
      # <await (nw == 0)
           nr = nr + 1; >
      P(e);  # Pick up baton
      if ( nw > 0 or dw > 0 ) {
         dr = dr + 1;
         V(e); P(r);
      }
      nr = nr + 1;
      if ( dr > 0
           && dw == 0) {
         dr = dr - 1;
         V(r);
      }
      else
         V(e);
      read the database;
      # < nr = nr - 1; >
      P(e);  # Pick up baton
      nr = nr - 1;
      if ( nr == 0 and dw > 0 ) {
         dw = dw - 1;
         V(w);
      }
      else
         V(e);
   } # while
} # Reader
```

```
process Writer[j = 1 to N] {
   while ( true ) {
      # < await (nr == 0 and
                 nw == 0)
              nw = nw + 1; >
      P(e);  # Pick up baton
      if ( nr > 0 or nw > 0 ) {
         dw = dw + 1;
         V(e); P(w);
      }
      nw = nw + 1;
      V(e);
      write the database;
      # < nw = nw - 1; >
      P(e);  # Pick up baton
      nw = nw - 1;
      if ( dw > 0 ) {
         dw = dw - 1;
         V(w);
      }
      elseif ( dr > 0 ) {
         dr = dr - 1;
         V(r);
      }
      else
         V(e);
   }
}
```

b.)

```
int nr = 0, nw = 0
sem e = 1, r = 0, w = 0;
int dr = 0, dw = 0;
int ReaderCount = 0;
```

```
process Reader[i = 1 to M] {
   while ( true ) {
      # <await (nw == 0)
           nr = nr + 1; >
      P(e);  # Pick up baton
      if ( nw > 0 ||
         (ReaderCount >= 5 &&
              dw > 0)) {
         dr = dr + 1;
         V(e); P(r);
      }
      nr = nr + 1;
      ReaderCount++;
      if ( dr > 0 &&
         (ReaderCount < 5 ||
              dw == 0)) {
         dr = dr - 1;
         V(r);
      }
      else
         V(e);
      read the database;
      # < nr = nr - 1; >
      P(e);  # Pick up baton
      nr = nr - 1;
      if ( nr == 0 and dw > 0 ) {
         dw = dw - 1;
         V(w);
      }
      else
         V(e);
   } # while
} # Reader
```

```
process Writer[j = 1 to N] {
   while ( true ) {
      # < await (nr == 0 and
                 nw == 0)
            nw = nw + 1; >
      P(e);  # Pick up baton
      if ( nr > 0 or nw > 0 ) {
         dw = dw + 1;
         V(e); P(w);
      }
      nw = nw + 1;
      ReaderCount = 0;
      V(e);
      write the database;
      # < nw = nw - 1; >
      P(e);  # Pick up baton
      nw = nw - 1;
      if ( dr > 0 ) {
         dr = dr - 1;
         V(r);
      }
      elseif ( dw > 0 ) {
         dw = dw - 1;
         V(w);
      }
      else
         V(e);
   }
}
```