Instructor: Pawel Gburzynski

# CMPUT 379, Fall 2000, Midterm Examination (A2)

Id number .....

Surname .....

Given names .....

This is an open book (and notes) exam. All questions have equal weights and must be answered in whatever space is available on this form. No additional sheets are allowed. Copying random excerpts from your class notes is not a good practice.

The exam is marked out of 20 which is the percentage of its contribution to the final mark.

## Question 1

Below is Attempt 4, as discussed in class, generalized for up to 32 processes. Each process knows its number (between 0 and 31) stored in the integer local variable `pnum`. The other variable used in this setup, `trying`, is a global integer (initialized to 0) storing the set of processes that are currently trying to enter the critical section. Its role is to avoid having to check 31 separate variables in the `while` condition. This is the generic code run by every process:

```
process ( ... ) {
  ...
  while (1) {
    ...
    trying |= (1 << pnum);
    while ((trying & ~(1 << pnum)) != 0) {
      trying &= ~(1 << pnum);
      rndwait ();
      trying |= (1 << pnum);
    }
    ...
    critical;
    ...
    trying &= ~(1 << pnum);
    ...
  }
}
```



This solution may work or it may not, depending on some properties of the underlying CPU architecture and, possibly, the compiler. What are those properties?

**Note:** the operators used in the above code are standard C operators, i.e., "`a << b`" means "shift `a` logically `b` bit positions to the left", "`a |= b;`" means "or `b` bitwise into `a`", "`a &= b;`" means "and `b` bitwise with `a` storing the result in `a`", '`&`' is bitwise "and", and '`~`' is bitwise negation. Integer numbers on the machine in question are at least 32 bits long.

## Question 2

One problem with the circular buffer is the ambiguity of its full/empty status. If we assume that `in==out` means that the buffer is empty, then one location in the buffer is going to be wasted. One smart programmer proposed to use a variable, named `Count`, to simply tell the number of items currently stored in the buffer. The buffer is empty if `Count==0` and full if `Count=N`. Give at least one good reason why this addition does not fit the concept of circular buffer very well.

# Question 3

A certain system uses resource ordering to prevent deadlock. The total number of resource units is M, the number of resource groups is G and the maximum size of a single group is S. Which of the following statements are true?

1. The maximum number of resource units that a single process can possibly hold is G.
2. A process can issue at most G requests for resources unless it releases some of those it is holding.
3. A process can never request more than S resource units at a time.
4. A process that is holding at least one resource from the highest-order group cannot request any more resources, even if they are freely available in lower groups.
5. No more than M/G processes can be holding some resources at the same time.

Explain all your selections.

# Question 4

One unpleasant property of the FIFO replacement strategy is that it tends to victimize permanently needed pages. Suppose that you have no "referenced" bit in hardware and, for some reason, you do not feel like emulating it in software. How would you try to modify FIFO, without resorting to the "referenced" bit, to avoid victimizing permanently needed pages?