

Operating Systems

Last Exam of the Millennium!

Name: _____

17 December 1999

1. Synchronization

a. Below is a partial solution to the dining philosophers problem. based on Christian's idea of a semaphore for each philosopher. Unfortunately, I have left one procedure, **put-forks(i)**, out. Give the body for the missing function and explain why the **test** function must be written as it is. You may assume that the syntax of the solution given so far is correct.

```
#define N 5      /* number of philosophers */
#define THINKING 0 /* phil is thinking */
#define HUNGRY 1 /* phil is hungary */
#define EATING 2 /* phil is eating */

/* shared variables */
int state[5] /* array to keep track of everyone's state */
semaphore mutex = 1; /* mutal exclusion for CS */
semaphore s[5]; /* an array of semaphores */

void take_forks(int i)
{
    P(mutex);
    state[i] = HUNGRY;
    test(i);
    V(mutex);
    P(s[i]);
}

void put_forks(int i)
{
    /* your code here. A solution is possible in 5 lines */

}

void test(i)
```

```

{
    if (state[i] == HUNGRY && state[(i - 1) mod 5] != EATING
        && state[(i+1) mod 5] != EATING)
    {
        state[i] = EATING;
        V(s[i]);
    }
}

philosopher(int i){
    while (TRUE){
        // Think
        take_forks(i);
        // Now you can eat
        put_forks(i)
    }
}

main( )
{
    for (int i = 0; i < 5; i++)
        philosopher(i);
}

```

b. In class we demonstrated a solution to the general semaphore problem using the test-and-set hardware. Assume that we had a different instruction named **Swap**. The swap instruction is a hardware instruction that executes as an atomic instruction. Its definition is:

```

procedure Swap(boolean a,b)
{
    boolean temp;
    temp = a;
    a = b;
    b = temp;
}

```

Give a solution to the CS problem using this hardware instruction.

2. Virtual Memory. A page-replacement algorithm should minimize the number of page faults. We can do this minimization by distributing heavily used pages evenly over all of memory, rather than having them compete for a small number of page frames. We can associate with each page frame a counter of the number of pages that are associated with that frame. Then, to replace a page, we search for the page frame with the smallest counter. Once a page has been assigned to a frame, it will always be swapped into that same frame, ie it cannot be placed into a different frame.

a. Define a page-replacement algorithm using this basic idea. Specifically address the problems of **1.** what the initial value of the counters is, **2.** when counters are increased, **3.** when counters are decreased and **4.** how the page to be replaced is selected.

b. how many page faults occur for your algorithm for the following reference string, for four page frames?

1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2.

c. What is the minimum number of page faults for an optimal page-replacement strategy for the reference string in part b with four page frames? With **LRU**?

3. Paging. Consider a paging system with two page sizes, 1KB and 4KB. The goal is to eliminate internal fragmentation as much as possible.

Assume that we have 64 bit addresses and that the physical address space is 4GB.

Answer the following questions about this imaginary architecture:

a. What is the max number of pages in terms of a power of 2 if all pages are 1KB in size? If all pages are 4KB? How many bits will be needed in the virtual address to represent a page?

b. How large (in terms of a power of 2) will a processe's page table be? How will the bits from the page address be used to access the page table? How can you handle the fact that there are two page sizes in the page table?

c. What is the max number of frames in terms of a power of 2? How many bits will be needed in the physical address to represent a page?

d. How large (in terms of a power of 2) will a processe's frame table be?

e. Draw two diagrams that starts with a logical address and shows how bits are distributed to access the page table and form a physical address. The first diagram should be similar to figure 8.26 in the textbook and the second diagram should be similar to figure 8.27. The first diagram shows how hardware is used to access memory, the second shows just the address translation. **Remember** to label all addresses with the number of bits being used.