NAME: _____ **SCORE:** _____

COSC 4330 FINAL EXAMINATION MAY 8, 2002

CLOSED BOOK. YOU CAN HAVE ONE PAGE OF NOTES. CHEATERS WILL BE EXPELLED FROM UH

1. Answer in one or two sentences to the following questions (6×5 points).

a) Give one example of a ***consumable resource*** in a distributed computing system.

b) What is the main disadvantage of having ***TLB misses*** handled by the kernel?

c) What is the purpose of the ***valid bit***? (5 points)

d) Give one example of a ***fixed-size*** access control list.

e) Is it possible to prevent deadlocks in a ***client/server system***? Why?

f) What can be done to fight ***external fragmentation***?

2. A computer has 512 megabytes of main memory, 32 bit addresses and a page size of eight kilobytes.

a) How many page frames are there in main memory? (5 points) _____ **frames**

b) How many bits of the virtual address remain ***unchanged*** by the address translation process? (5 points)

_____**bits**

c) On the average, how much memory is lost to ***internal fragmentation***? (5 points)

_____ **bytes per process**

3. A Berkeley UNIX file system has 16 kilobyte blocks. How many ***blocks*** of any given file can be accessed**:**

a) Directly from the i-node (5 points): _____ **blocks**

b) With one level of indirection (5 points): _____ **blocks**

c) With two levels of indirection (5 points): _____ **blocks**

4. Describe the Mach page replacement policy. (10 points for answer including a correct diagram) How does it compare with the VMS policy? (2×5 points: there is one main advantage and one main disadvantage)

5. A small parking lot has a single entrance and just enough space to handle ten cars. Add the required pseudocode to the following monitor to ensure the smooth operation of the lot. Your solution should prevent cars from entering the lot when it is full. It does not need to consider the case of people trying to remove a car from an empty lot. (4×5 points)

```
Class parking_lot {
     private free_spaces; // number of free spaces
     private condition not_full;

     public void synchronized park() {
          _____________
          _____________
          enter_the_lot_and_park_your_car()
          _____________
          _____________
     } // park()

public void synchronized remove() {
          _____________
          _____________
          pick_up_your_car_and_leave_the_lot()
          _____________
          _____________
     } // remove()

parking_lot() {
          free_spaces = ________;
     } // constructor

} // Class parking_lot
```