

Programming Languages and Systems
Comprehensive Exam, Spring 2002

Yale University

May 20, 2004

First Name: _____

Last Name: _____

Yale NetID: _____

Instructions

1. This exam contains **6** problems, solve all of them. Do not spend all your time on a single problem; if you get stuck, move on. Some questions are vague on purpose, please state your understanding, and then explain your answer based on your assumptions.
2. The exam is “nearly open book.” You may bring textbooks from the reading list, plus one page of notes.
3. Answer each question on a separate piece of paper so that different faculty members can grade different questions in parallel. Try to write the solution in the space provided with each problem. You can use the back side of each sheet if necessary.
4. Write your name on top of each page so that the individual sheets can be separated.

Problem	Topic	Points
1	Computer Architecture	
2	Operating Systems	
3	Concurrency	
4	Programming Languages and Compilers	
5	Programming Languages	
6	Computer Networks	
TOTAL		

Problem 1. Computer Architecture (10 points)

Assume you have a 64-byte cache with 8 bytes per block. The memory is byte addressable. All cache blocks start out as invalid. For (a) through (c) below, assume that the cache is direct mapped. Answer the following questions using the following address stream:

Addresses: 80, 111, 60, 94, 112, 35, 60, 45, 112, 10, 70, 15, 80, 112, 120, 60, 90.

The address “80” is at position 1, the address “111” is at position 2, etc.

(a) How many cache hits are due to temporal reuse, and which address references are they (give the position in the address stream)?

(b) How many cache hits are due to spatial reuse, and which address references are they (give the position in the address stream)?

(c) How many cold start (compulsory) misses are there and give their position in the address stream?

Addresses: 80, 111, 60, 94, 112, 35, 60, 45, 112, 10, 70, 15, 80, 112, 120, 60, 90.

(d) Now assume that the cache is 4-way associative with LRU replacement policy. Which addresses (give the address and position in the address stream) are cache hits? Label the hits as either spatial or temporal.

(e) Now assume that the cache is fully associative with LRU replacement policy. Which addresses (give the address and position in the address stream) are cache hits? Label the hits as either spatial or temporal.

Problem 2. Operating Systems (10 points)

(a) When writing a new block to a file, three on-disk things need to be updated. These are:

- 1) Write the new file data to the data block.
- 2) Get the disk block from the free list.
- 3) Update the inode (file descriptor).

In what order must these operations be done to be able to safely recover from a system crash?

(b) When an operating system crashes, it often leaves the data that it stores on the disk in an inconsistent state. By inconsistent state, we mean that file system data structures do not correctly describe which blocks should be in the files, descriptors (inodes), and free lists. This can happen because it takes several disk reads and writes to update a file, its inodes (file descriptors), and the free list. Remember that multiple processes and users can be updating their files in the same time interval.

When the operating system is rebooted, it is necessary to validate that the file system and its data structures are all right (or figure out how to fix them). “Consistent state” means that all blocks are either in the free list or in some inode, and that all blocks contain valid data.

Assume that you have an allocation scheme based on UNIX inodes and that free space is described by a block bit map.

After a crash, describe the algorithm used at boot time to determine if the file system is in a consistent state.

Problem 3. Concurrency

Below is code that solves the classic Dining Philosopher problem such that it is live as well as safe; this is done with the addition of a state variable for each philosopher, designating whether they are THINKING, EATING, or HUNGRY. It uses Semaphores that support the traditional **P** and **V** operations.

```
Semaphore mayEat[5]; // How is this initialized?
Semaphore mutex;    // How is this initialized?
int state[5];      // Initialized to THINKING
int p;             // Initialized to a unique id

void take_chopsticks() {
    mutex.P();
    state[p] = HUNGRY;
    test(p);
    mutex.V();
    mayEat[p].P();
}

void put_chopsticks() {
    mutex.P();
    state[p] = THINKING;
    test((p+1)%5);
    test((p+4)%5);
    mutex.V();
}

void test(int i) {
    if (state[i]==HUNGRY && state[(i+1)%5]!=EATING && state[(i+4)%5]!=EATING) {
        state[i] = EATING;
        mayEat[i].V();
    }
}

void run() {
    while (1) {
        Think();
        take_chopsticks();
        Eat();
        put_chopsticks();
    }
}
```

Each philosopher is a separate thread executing the run() method. Answer the following questions:

(a) How should the Semaphore elements of `mayEat` be initialized?

(b) How should the Semaphore mutex be initialized?

(c) What is the maximum number of Philosophers that can be waiting on a Semaphore element `mayEat[i]` at any given time?

(d) What is the maximum number of Philosophers that can be waiting on mutex at any given time?

(e) Does the code work correctly if the statement `mayEat[i].P` is moved before `mutex.V()` in `take_chopsticks()`? Briefly describe why or why not.

(f) Does the code work correctly if the statement `mayEat[i].V` is moved before `state[i]=EATING` in `test()`? Briefly describe why or why not.

(g) Does the code work correctly if the statements `test((i+1) % 5)` and `test((i+4) % 5)` are moved before `state[i]=THINKING` in `put_chopsticks()`? Briefly describe why or why not.

Problem 4. Programming Languages and Compilers (10 points)

(a) Garbage Collection was initially developed to support higher-order functions for the functional language Lisp, but it is now widely adopted by modern type-safe languages such as Java and C#. Explain why higher-order functions would force the use of garbage collection.

(b) A higher-order function is represented as a closure at runtime. Two commonly used representation strategies are called flat closures and linked closures. Given the following ML code, show how closures S and T are represented at runtime under each strategy.

```
fun ilist(n) =
  if n <= 0 then [] else n::(ilist(n-1))

fun ihead (a::r) = a
  | ihead [] = 0

fun P(v, w, x, y) =
  let fun Q() =
        let val u = ihead(v)

            fun R() = (u, w+x+y+3)
              in R
            end
        in Q
      end

  val S = P (ilist(1000), 0, 0, 0)
  val T = S()
  val result = T()
```

Your Name:

Yale NetID:

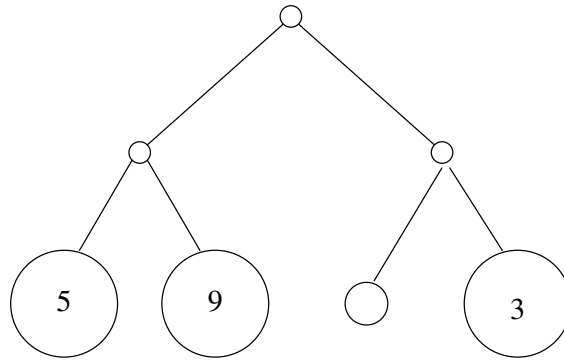
(c) Does Java support higher-order functions? How do they differ from ML-style higher-order functions? Is there a way to implement ML-style higher-order functions in Java?

Problem 5. Programming Languages (10 points)

In this problem you are asked to explore a function that ranges over trees of the form

```
datatype 'a Tree
  = Nil
  | Node of 'a Tree * 'a Tree
  | Leaf of 'a
```

An example of a tree is given by the following picture.



The medium sized circles correspond to `Nil`, the small circles to `Node`, and the large circles to `Leaf` nodes, that can carry objects of any type. In ML we would represent this tree as

```
val t = Node (Node (Leaf 5, Leaf 9), Node (Nil, Leaf 3))
```

Consider the following function, an *iterator* on trees, which expects four arguments. The first three correspond to actions to be taken, if a `Nil`, `Node`, or `Leaf` is encountered, and the fourth is the subject of the iteration, a tree.

```
fun iter f g h Nil = h
  | iter f g h (Node (t1, t2)) = f (iter f g h t1, iter f g h t2)
  | iter f g h (Leaf t) = g t
```

(a) Give the most general type for `iter`. *Hint: The function is polymorphic. Be careful with the placement of parentheses.*

(b) Write a function that negates all numbers stored in the Leafs of an `int Tree`, using nothing else but the function `iter` applied to some arguments. Do *not* define it by cases.

```
negate : int Tree -> int Tree
```

(c) Write a function that computes the sum of all numbers stored in the leafs of an `int Tree`, using nothing else but the function `iter` applied to some arguments. Do *not* define it by cases.

```
sum : int Tree -> int
```

(d) Write a function that computes a list of all numbers stored in the leafs of an `int Tree`, using nothing else but the function `iter` applied to some arguments. Do *not* define it by cases.

```
flatten : int Tree -> int list
```

(e) Show the following property for all trees t of type `int Tree` formally.

$$\text{sum (negate } t) \equiv \sim (\text{sum } t)$$

Prove this property by induction. State clearly what you are inducting over, what the induction hypotheses are, when you use them, etc.

Problem 6. Link Layer Services of Computer Networks (10 points)

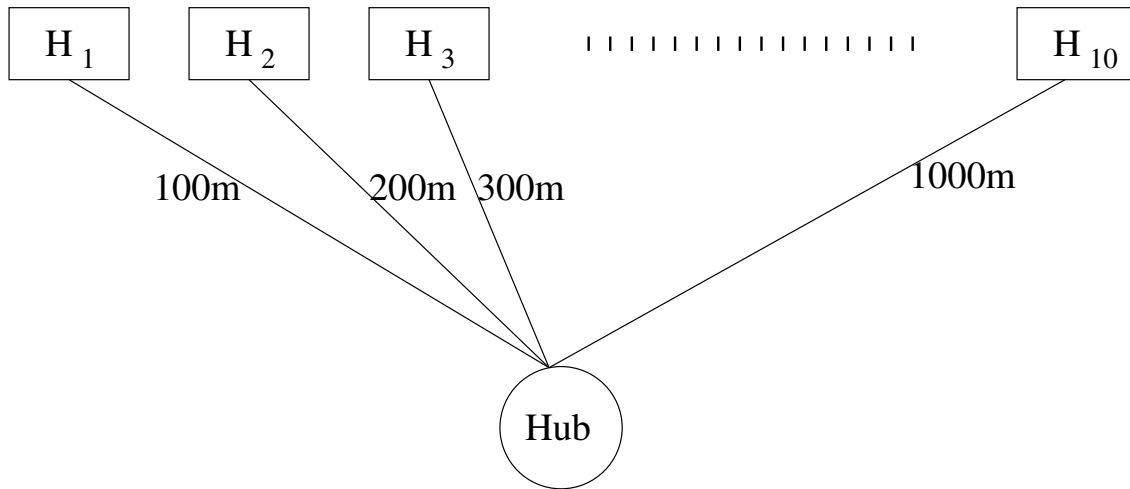
The link layer of a computer network provides many services. Below we consider two of them: error detection/correction and media access control.

(a) Many link layer protocols use error correction codes if bit error rates are high. At least how many check bits do you need in order to correct any single bit error for a packet with 500 bits?

(b) If bit error rates are not too high, it is more important to just detect bit errors and then request retransmission. One commonly used error detection code is Cyclic Redundant Check (CRC). Consider the CRC generator polynomial $G(x) = x^8 + x^7 + x + 1$. Assume that the data frame is 101101011010. What will be the transmitted bits, including both data bits and CRC check bits?

(c) The designer of the $G(x)$ above claims that the $G(x)$ can detect any bit errors if the number of error bits is an odd number. Is this true or false? Please justify.

Another service of the link layer is media access control (MAC). Among all of the MAC protocols CSMA/CD is the most widely used. For the questions below we assume that the speed of propagation of a signal along a cable is 2×10^8 m/s. The figure below shows a 10 Mbps CSMA/CD network interconnecting ten computers. Each computer is connected to the hub with a cable of different length. Computer H_1 is connected via a 100 m cable, H_2 computer via a 200 m cable and so on, up to computer H_{10} , which is connected via a 1000 m cable (ignore the requirement of repeater due to signal degradation).



(d) Calculate the shortest packet length L_{min} of this network so that CSMA/CD protocol will function correctly.

(e) Does a CSMA/CD network require a maximum packet length in order for the protocol to work correctly? If so, what is the maximum?

Your Name:

Yale NetID:

(f) If the network carries packets that are all 1000 bits long, please calculate the approximate efficiency (Reminder, efficiency is the percentage of bandwidth that is used to transfer useful data. For CSMA/CD, the average number of collisions before a successful transmission is approximately 5).

(g) If all of the links are made 10x longer and the rate of the network made 10x faster, how much larger must L_{min} become compared to your answer in (d)?