Name _____ Email _____ ID Code _____

# Systems Breadth Exam

19 February 2002

This is a **closed-book**, **closed-notes**, written examination. You must answer *all* questions in the "Core" section. You will then answer all questions in the other two sections you have chosen. You have **three hours**.

Do not write any answers on this paper. Write **only** your *name*, *ID number*, and *assigned ID code* on this paper. Write your answers on the blank sheets provided. Use only one side of each answer sheet. **Write your assigned ID code and the page number at the top of each answer sheet.** In particular, do *not* put your name on your answer sheets.

# 1 Core

1. The function EQ3 is a 3-bit comparator: it takes two 3-bit inputs and outputs a 1 if they are equal and 0 otherwise.

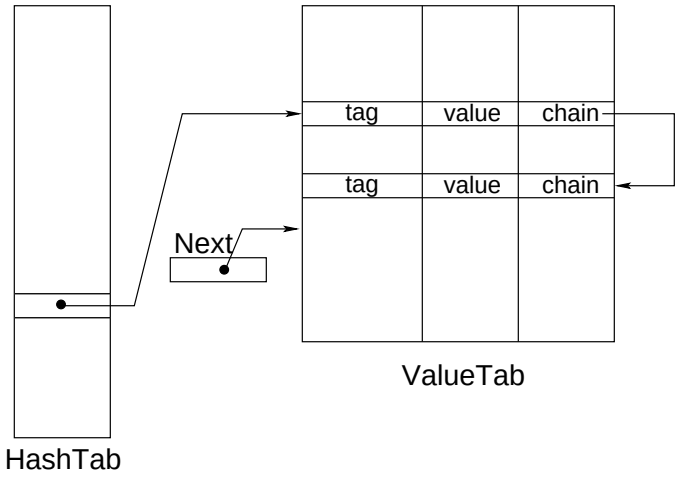
   a. Write down boolean functions to describe the output of EQ3 in terms of its inputs $A_0$, $A_1$, $A_2$, and $B_0$, $B_1$, $B_2$. Your answer should be in minimal sum-of-products form.

   b. Draw a circuit diagram for your implementation, using only NAND gates.

   c. Describe how you would implement your function using a PROM, that is, a programmable read-only memory. ("Programmable" means you can put whatever you want into the locations of the memory, and it won't change once you put it there.) How many bits of PROM memory are required to implement EQ3?

2. It is not unusual for a computer today to have a gigabyte or more of "real" memory. With that much storage, almost all programs can fit entirely in the machine's memory. Yet, even if all programs running programs could fit simultaneously in the real memory, *virtual* memory would still be a useful mechanism. Explain why.

3. The figure below shows a data structure used in implementing an *associative store*. The store allows *values* to be stored and retrieved by associating them with user-selected *tags*. At all times, there is at most one entry that contains any given tag.




   The hash table contains indices into the value table. The value table contains records of type ENTRY, which have the following fields:

```
TAG   tag;
VALUE value;
INDEX chain;
```

   Entries whose tags hash to the same value are chained together through their `entry_chain` fields; this field contains the index of the The end of the list is indicated by special index END. The variable *Next* contains a pointer to the next free (unused) entry in the value table.

   The store is accessed through two methods *get* and *put*:

- `VALUE get(TAG t)`—returns the value associated with the tag $t$, if there is one, and the special result `NIL` (which is guaranteed to be different from every meaningful value) if there is no value associated with $t$ in the store. Get is implemented as follows:

```
h = hash(t);
current = HashTab[h];
while (current != END AND ValueTab[current].tag != t)
   current = ValueTab[current].chain;
if (current == END)
   return NIL;
else
   return ValueTab[current].value;
```

- `void put(TAG t, VALUE v)`—associates the value $v$ with the tag $t$. If some value was bound to the tag $t$ prior to the call, it is overwritten. If nothing was bound to the tag $t$ before the call, a new entry is created. The *put()* method is implemented as follows:

```
h = hash(t);
current = HashTab[h];
last = END;
while (current != END AND ValueTab[current].tag != t) {
   last = current;
   current = ValueTab[current].chain;
}
if (current == END) {  // not found; allocate it
   if (last == END) // hash table slot was empty
      HashTab[h] = Next;
   else
      ValueTab[last].chain = Next;
   current = Next;
   Next = Next + 1;
}
ValueTab[current].tag = t;
ValueTab[current].value = v;
ValueTab[current].chain = END;
```

We ignore the question of how bindings are removed from the store.

a. The above code works fine for a single thread, but has problems if there are multiple threads and their execution is interleaved (i.e. calls to *get()* and *put()* are not atomic). Explain the problem and give an example of what can go wrong.

b. Show how to correct the problem. Use any approach you want, but your solution must allow execution of *get()* and *put()* to be interleaved—that is, you may not simply make the methods atomic. If you assume the existence of any synchronization primitives, explain their semantics. (Note: simpler is better.)

## 2 Databases

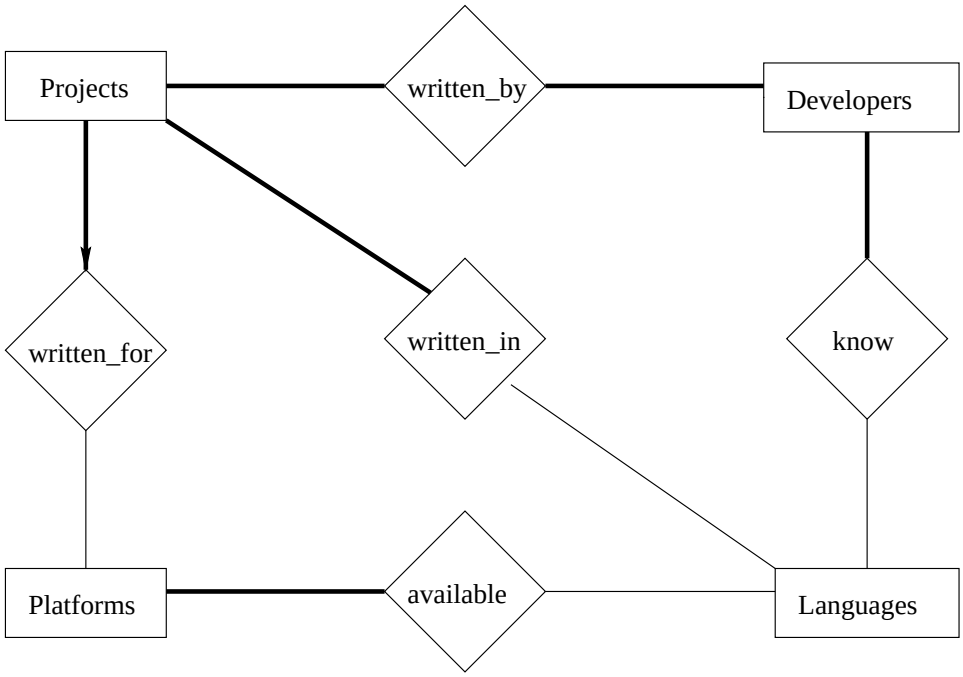



Figure 1: E-R Diagram.

1. Consider the Entity-Relationship diagram describing the operations of a software company (Figure 2). Let the entity sets in the diagram have the following attributes:

   Projects(Id, CodeName, StartDate, EndDate, Budget);
   Developers(SSN,Name, StartDate, Position, Salary);
   Languages(Id, Name, Type);
   Platforms(Id, Hardware, OS);

   a. What participation and key constraints are present in the diagram? Describe the meaning of each constraint in plain English.
   b. Translate the E-R diagram to the relational model. Write all necessary SQL commands to create the approprate relations.

2. For the relational model that you built in Problem 1, express the following queries in SQL:

   a. Find all developers who work on projects written in Perl. Return names and salaries of the developers.
   b. Find the total of the salaries of all developers who work on projects for Macintosh (hardware) OS X (software) platform.

c. Find all developers who do not know Java and work on projects for which Java is used. Return the names of the developers and the codenames of projects.

d. Find total budgets for projects by platform. For each platform report its hardware and software component, total number of projects under development and the sum of budgets of all these projects.

3. Write query (a) from the previous problem in relational algebra.

4. Consider the following set of transactions:

| T1 | T2 | T3 | T4 |
|---|---|---|---|
| R(A) | | | |
| | R(A) | | |
| | | R(B) | |
| | | | R(C) |
| W(B) | | | |
| | W(C) | | |
| | | W(A) | |
| | | | W(C) |
| commit | | | |
| | commit | | |
| | | commit | |
| | | | commit |

a. Indicate all blind writes in the schedule above.

b. Is the schedule above serializable? Why or why not? (Show your work).

c. Rewrite the schedule using 2-Phase locking with deadlock prevention using *wait-die*. Transactions that have started earlier have higher priority (e.g., at the beginning T1 has priority over T2 which has priority over T3 which has priority over T4).

## 3 Programming Languages

1. Your boss has heard that object-oriented languages are hot, and that lots of folks use Prolog. You are told to design object-oriented prolog (to be called Orlog). You complain that this combination of ideas makes no sense. Your boss tells you to explain why, in writing. That explanation is the first part of this problem.

2. Your boss wasn't convinced, and maybe didn't even read your explanantion. You are requested to design Orlog even though you think it is silly. So you do your best to connect the ideas of object-orientation and Prolog. Your design of Orlog is the second part of this problem.

# 4 Compilers

Consider the following extended form of the **while** loop (in fact there is an experimental programming language using it) provided in the EBNF notation where the part in "[, ]" is repeated zero or more times.

WhileStatement ::= "while" ⟨Expression⟩ "do" ⟨Statement⟩
[ "else" ⟨Expression⟩ "do" ⟨Statement⟩ ]

An example WhileStatement is:

**while** (b == true) **do** { i = i + 1; }
**else** (j == 2002) **do** { i = i + 2; }
**else** (j > 2500) **do** { i = 1000*i; }

The semantics of WhileStatement are that ⟨Expression⟩s are evaluated one at a time in the given order until one of them is found to be true. The corresponding ⟨Statement⟩ is then executed and the process of evaluating conditions (⟨Expression⟩s) is repeated. If every ⟨Expression⟩ evaluates to false, the loop terminates.

Assume that you want to generate code for a stack machine (P-stack computer) You may assume that the P-stack machine uses the following instructions "U_Jump L" for unconditional jump to label L and "Jump_on_True L" for a jump to L on 'true'. The generated code is stored in the computer memory addresses as an array. For example, the cell with address 200 is referred to as mem[200].

1. Describe (or draw a picture of) the schematic view of code generated for WhileStatement production.

2. Attribute this production with semantic actions (to make an attribute grammar for a syntax directed translation) to handle code generation. You may assume that you are given function '`emit_code()`', and that the calls `emit_code(Expression)` and `emit_code(Statement)` generate correct code for `Expression` and `Statement`, respectively. Also, `emit_code(jump L)` produces code `jump L`. You will need other functions as well; please define them. If it is convenient for you, you can rewrite the production in the BNF notation.

# 5 Distributed Operating Systems

1. Write a monitor-based solution for the following version of the readers-writers problem:
   *At most 10 readers can be in the critical section at any given time. Writers have priority over readers, that is, a waiting or arriving reader gains access only when there are no writers in the system.*

2. State whether the following statements are true or false. **Justify your answer, i.e. if the statement is true prove it; if it is false give a counter example**.

   **Notation:** If $a$ denotes an event of a distributed computation, $l^a$ denotes the Lamport timestamp of event $a$, and $t^a$ denotes the vector timestamp of event $a$. $\rightarrow$ denotes Lamport's "happened before" relation among events. $\|$ denotes concurrency among events).

   a. If $a \| b$ and $b \| c$ then, $a \| c$.
   b. If $a$ and $b$ are two events of a distributed computation, then one of the following statements is always true
      - $t^a < t^b$
      - $t^a = t^b$
      - $t^a > t^b$
   c. If channels are FIFO then messages are causally ordered
   d. If $l^a < l^b$ then $a \rightarrow b$
   e. Lamport's mutual exclusion algorithm will work correctly if channels are not FIFO

3. Discuss the advantages and disadvantages of the following in the context of distributed file systems

   a. full-file caching and block file caching
   b. stateless file server and stateful file server
   c. client cache in memory and client cache in disk

4. What are the main causes of thrashing in distributed shared memory systems? How does page size affect the performance of a DSM system?

5. What is two-phase locking? What problem can it cause? How does *strict* two-phase locking eliminate this problem? What is the drawback of strict two-phase locking?

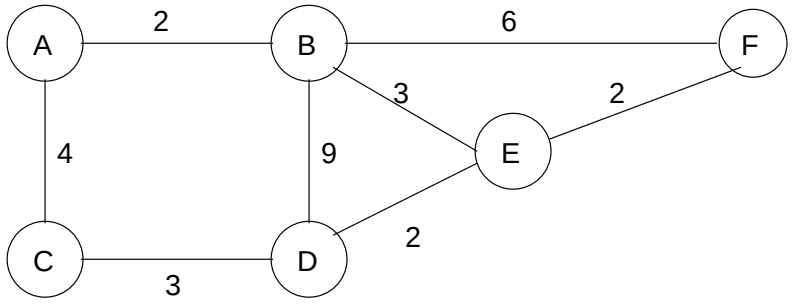
# 6 Networks

1. Suppose you want to transmit some data on a wire using the byte 0x6E (01111110) as a frame delimiter. However, it is possible that the pattern 0x6E may occur as one of the bytes in the message. How can you prevent the receiver from prematurely thinking the frame has ended? Illustrate your scheme with an example message containing the byte 0x6E.

2. Consider a stop-and-wait protocol being used on a 100Mbps ($10^8$ bits/sec) point-to-point link. The one-way latency of the link is 5 milliseconds (0.005 sec) in each direction, and the maximum frame size is 1250 bytes.

   a. How long will it take to transmit a file of 1.25 MB ($1.25 \times 10^6$ bytes) using this protocol? You may assume no losses occur, that processing takes no time, and that acks are zero bits long.

   b. Which change would have the greater effect on performance: cutting the latency in half (to 5 milliseconds round-trip), or doubling the bit rate to 200 Mbps?




3. This problem deals with routing in the network shown above, which is represented as an undirected graph. The numbers next to edges represent distances, and shortest-path routing is in use.

   a. A long time after the network starts up, show the routing table at Node B.

   b. Now suppose the link between B and E goes down. Explain what happens (i.e. what messages are sent where, and what tables get updated) if a *distance-vector* protocol is in use.

   c. Repeat for a link-state protocol.

4. In the TCP/IP protocol suite, the *port* is the only form of address defined above the network layer. However, ports are only meaningful in the context of a network address. In other words, it is necessary to give both an Internet address and a port number to uniquely identify a particular program.

   a. Describe a scheme where this is not the case, that is, where transport addresses can be interpreted independently of any network address.

   b. What advantages might be realized if your scheme were implemented?

   c. What problems might there be in implementing this scheme?