<table>
<thead>
<tr>
<th>Page</th>
<th>Possible</th>
<th>Score</th>
<th>Graded By</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

System Software
Ph.D. Qualifying Exam

27 January 1998

EXAMINEE #

1. This exam is written with 100 points.

2. The exam contains 11 pages. Please check to be sure your copy of this exam contains pages 1 through 11, and problems 1 through 6.
1. Constant Propagation (15 pts)

Suppose we have a multi-pass compiler that includes passes for constant propagation, dead code elimination, register allocation, useless code elimination, and identifier resolution.

(a) Which pass(es) would generally produce worse results if they preceded constant propagation than they would if they followed it? For each, explain why.

(b) Is there any benefit to performing the constant-propagation pass more than once? Why or why not?
2. Compiler Generators (10 pts)

Porting compilers to new platforms effectively is a somewhat labor-intensive task. As much as forty years ago, people wondered if we couldn’t simply specify compilers for a generic machine, and then feed this to a “compiler compiler” along with the specifications for a particular machine and have it generate a compiler for that machine. Why is this not common practice today?
3. Left Left (25 pts)

C++ has a left shift operator which looks like $\ll$. C++ also has a mechanism called templates, in which $<$ and $>$ are used as matching delimiters. Since templates allow these delimiters to be nested, we can legally write $A\ll B>C$, in which the $\ll$ must not be scanned as a left shift operator, but as two opening $<$ tokens. What problems does this pose for a C++ implementor, and how would you solve them?
4. Critical Sections (26 pts)

Our 501 textbook by Silberschatz and Galvin discusses the critical section problem in Chapter 6. A good solution to the critical section problem is said to satisfy *mutual exclusion*, *progress* and *bounded waiting*. The concept of progress is difficult to define. Silberschatz' definition includes “no indefinite waiting” and another condition which his first algorithm (the *Turn Algorithm* below) violates. For our purposes here we define the latter concept as progress and separate *indefinite waiting* from the definition of progress:

Definition *Progress*: Suppose the processes that have exited their critical section have had the opportunity to so indicate and assume further that no process is in its critical section. Then if process $P_i$ is ready (requests) to enter its critical section and $P_i$ has its *cpu* and $P_i$ is next to leave any associated queue for entering the critical section, then $P_i$ enters.

So for the purposes of this qualifying exam question we assume that a good solution for the critical section problem satisfies (1) Mutual Exclusion, (2) Progress (as given here), (3) No Indefinite Waiting (or blocking), and (4) Bounded Waiting.

Consider the Turn algorithm in Figure 4 for two processes $P_0$ and $P_1$ and answer the related questions on the following pages.

```
j := (i+1)mod(2);
turn := 0;

repeat
  while (turn != i) do no-op;
  CRITICAL SECTION
  turn := j;
  REMAINDER SECTION
  until false;

exit;
```

Figure 0.1: Turn Algorithm Pseudo Code for $P_i$. 
(a) What is/are the shared variable(s) in the pseudo code for the Turn Algorithm?

(b) What would be the effect to initializing turn to 1 rather than 0?

(c) Explain carefully by example how progress (as defined here) can be violated with $P_1$ in its REMAINDER SECTION.

(d) Explain carefully by example how no indefinite waiting can be violated with $P_0$ being “busy waiting”.

(e) Does Bounded Waiting hold for the Turn Algorithm (yes or no): _______. If yes, what is the bound?

(f) The Turn Algorithm can be modified to accommodate three processes so that the order of entry into the critical section is $P_0$, $P_1$, $P_2$, $P_0$ and so forth. Then (fill in the blanks) rather than setting $j$ equal to the other process’ id ($turn := j$), we could set $turn := _______ \mod (______)$. 

Score on this page: __________
(g) Depict below (by pseudo code) how a binary semaphore, guard, can be used to protect a critical section so that mutual exclusion holds:

```
CRITICAL SECTION

The semaphore guard should be initialized to _____.

What can be associated with guard so that progress, no indefinite waiting and bounded waiting would be satisfied?

If there are n processes, what would then be the bound for bounded waiting?
```
5. Deadlock (17 pts)

(a) For each of the following situations, either draw a resource allocation graph with exactly two resource types satisfying the stated conditions, or state why the situation is not possible. If

i. A system that is unsafe but not deadlocked.
ii. A system with a cycle but no deadlock.
iii. A system with a deadlock but no cycle.
iv. A system where processes are blocked but there is no deadlock.
v. A system that is safe.
(b) Consider a system with 4 processes, \( p_1, \ldots, p_4 \); four reusable resource types, \( r_1, \ldots, r_4 \), and four units of each resource type (vector \([4, 4, 4, 4]\)).

The current system state is as shown below:

<table>
<thead>
<tr>
<th>Current Allocation</th>
<th>Maximum Allowable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0 1</td>
<td>2 1 2 2</td>
</tr>
<tr>
<td>0 2 1 1</td>
<td>1 3 1 1</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>2 0 2 1</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>0 1 0 1</td>
</tr>
</tbody>
</table>

i. What is the remaining claim? The current available vector?

ii. What is the system state? Explain.
iii. For each of the following give an example of a single request that satisfies the stated conditions.
   A. A legal request that leaves the system in an unsafe state.
   B. A legal request that cannot be granted at this time.
   C. A legal request that keeps the system state safe.
   D. An illegal request.
6. Virtual Memory (8 pts)

Consider a “hybrid” page table structure. The top-level structure, which is always in
memory when the process it belongs to is running, consists of 16 entries.

The first 13 entries are the entries for the first 13 pages of the process. The 14th
entry adds 1 level of indirection. It points to a page that contains the next page table
entries for the process. For example, if a page table entry consists of 1 byte, and
the page size is 50 bytes, then the 14th entry in the top-level structure points to a
page that contains the page locations of pages 14-53 of the process. The 15th entry
of the top-level structure has 2 levels of indirection, and the last entry has 3 levels of
indirection.

(a) If this hybrid structure is used in a system where each page is 1024 bytes, and
each page table entry is 32 bits, approximately how big is the virtual address
space? Explain.

(b) If we desire a 4 terabyte (4 x 2^{40} bytes) virtual address space, what is the minimum
page size needed (assuming it is a power of 2 and each page table entry is still 32
bits)? Explain.
(c) With a page size of 4096 bytes, and 32 bits per page table entry, approximately how much space can the entire page table structure occupy? What if each page table entry is only 16 bits?

(d) What is an advantage to using a hybrid page table such as the one described here?