Your Secret ID NUMBER: ________

Maximum score: 100 points

Question 1: A simple synchronization problem (12 points)

You have been hired to coordinate dancers from team red and team blue. Each dancer is modeled by a separate thread, executing:

```c
red() {
    redTeamMemberReady();
    dance();
}
blue() {
    blueTeamMemberReady();
    dance();
}
```

The teams each comprises an arbitrary, but perhaps different, number of members.

Your task is to implement the body of procedures `redTeamMemberReady()` and `blueTeamMemberReady()` to ensure dancers that execute procedure `dance()` are paired, where a set of dancers on the dance floor is paired if each dancer from team red can be associated with a unique dancer from team blue and vice versa. Your code should not block dancers unnecessarily, but dancers need not be treated fairly—a late arriving dancer is allowed to execute `dance()` before other dancers who have been waiting for some time.

Solutions should be written in C, JAVA, or some other programming-like notation. Imprecise solutions—especially those written in English—will receive no credit. Please note that the solution has two parts: 1.1 and 1.2. You must solve both parts to get full credit!

1.1. [6 points] Give code for the bodies of procedures `redTeamMemberReady()` and `blueTeamMemberReady()` using semaphores to implement any synchronization.
1.2. [6 points] Give code for the bodies of procedures redTeamMemberReady() and blueTeamMemberReady() using monitors with condition variables to implement synchronization. Assume the signal-delayed discipline (Mesa-style condition variables).
Your Secret ID NUMBER: ________

**Question 2: Relational Model (15 points total)**

Consider the following relational schema (primary keys are underlined):

- Customer(Cid, CName, Age)
- Buys(Cid, Pid, Qty)
- Product(Pid, PName, Price)

2.1. **[5 points]** Write the following query in SQL:

```
Find all the products that were bought by the customer with the highest age.
```

You can assume that no two customers have the same age.

2.2. **[10 points]** Assume that the Product relation has an unclustered B+ tree index on the Price attribute, and this is the only index on the Product relation. The B+-tree has height 4, and each leaf-level index page has 50 (key, rid) pairs on the average. The Product relation has 10000 tuples with 10 tuples per page. The values of the attribute Price is uniformly distributed in the interval [1,1000]. Now consider the following query:

```
SELECT P.Pid, P.PName 
FROM    Product P 
WHERE P.Price >= val
```

Compute the minimum value of $val$ such that the most efficient way of executing the above query is to use the B+-tree index. (It is sufficient if you write down an expression that needs to be solved for the minimum value of $val$.) Assume that the cost metric is the number of page I/Os. In your analysis, you can assume that the worst-case scenario in computing the cost of a range scan using an unclustered B+-tree index (i.e., retrieving each tuple requires a page I/O). Explain your answer.
Question 3: Concurrency Control, Recovery (20 points total)

3.1. Consider a database system where transactions always access data in a given order; so the data items can be ordered $X_1 < X_2 < \ldots < X_N$ such that if a transaction accesses data item $X_i$, it will not subsequently access any item $X_j$ with $j < i$. We call such a database system an ordered database system.

3.1a. [4 points] Show that in an ordered database system, we still need concurrency control by giving an example of a non-serializable schedule for this system.

3.1b. Assume that 2PL (but not necessarily strict 2PL) is the concurrency control mechanism in an ordered database system. Which of the following properties are guaranteed? Give a one-line justification for your answer.

1) [1.5 points] The resulting schedule is deadlock-free.

2) [1.5 points] The resulting schedule avoids cascading aborts.

3) [1.5 points] The resulting schedule is recoverable.

4) [1.5 points] The resulting schedule is conflict-serializable.
3.2. [5 points] State-of-the-art recovery algorithms like ARIES use a combination of undo and redo records in the log. As an alternative, consider a purely undo based recovery scheme: the log contains undo information, but no redo information. Describe the order in which I/O operations must be performed to ensure correctness, including both writes to the log and flushing of dirty blocks from the buffer cache to the disk.
Question 4 [23 points]: True/False and Multiple-Choice questions, 1 point for each correct answer. No (0) points if you skip a question. However, to discourage guessing, incorrect answers receive –1/2 pt. Your minimum grade on Q4 is 0.

For each question, circle the best answer. Unless otherwise noted, select a single answer for multiple choice questions. In True/False questions, T denotes “true” and F denotes “false”

4.1. The best reason to use semaphores rather than busy-waiting in a user-level program is:
   (a) decreases the chance of deadlock
   (b) decreases the chance of livelock
   (c) allows more different types of synchronization to be coded
   (d) avoids wasted processor cycles
   (e) avoids expensive context switches

4.2. Good semaphore programming style stipulates that the process performing a P on a semaphore always be the process performing the corresponding V on that semaphore. T / F.

4.3. Good semaphore programming style stipulates that the process performing a P on a semaphore never be the process performing the corresponding V on that semaphore. T / F.

4.4. When using the “Banker’s Algorithm” for resource allocation, a process that requests a resource may be forced to wait even if there are far more units available than it requests, and even if no other process currently holds even a single unit of that resource. T / F.

4.5. When using the RSA public key algorithm, no problem arises analogous to the need to distribute symmetric keys, as seen (for example) when using DES. T / F.

4.6. People do not program optimal page replacement algorithms (such as the Dynamic paging policy called OPT or “Belady’s” OPT) today because
   (a) it is impossible to do so
   (b) it is too expensive to build the hardware support
   (c) page replacement costs are not sufficiently high for optimality to be a significant concern
   (d) there exist better-performing page replacement algorithms
   (e) The OPT policies are only applicable in computers that support page reference bits within the PTE. Not all processors do so.

4.7. Four out of the five of these instantiate a single basic principle. But the fifth one is essentially unrelated. Which is the unrelated one?
   (a) processor (L2) cache
   (b) disk buffer pool
   (c) page table
   (d) translation look aside buffer
   (e) DNS cache

4.8. LRU (least recently used) is a “stack” page replacement algorithm and hence is not subject to Belady’s anomaly. T / F.

4.9. Working set algorithms are used primarily in settings where the application knows what pages it will access, and can use this to provide hints to the paging subsystem. T / F.

4.10. Which of the following abstractions are implemented entirely in the kernel of an operating system such as Linux or Windows:
   (a) remote object invocation and remote procedure call
   (b) the process abstraction
   (c) dynamically linked libraries
   (d) the window manager
   (e) public key encryption/decryption
4.11. When monitors are used for synchronization and access to shared data:
(a) deadlock becomes impossible
(b) race conditions on access to that data become impossible
(c) livelocks become impossible

4.12. Which of the following are not typically saved when a process issues a system call that blocks, causing a context switch to some other process. (Circle all that are not saved)
(a) contents of registers
(b) program counter value
(c) values of condition-code flags
(d) processor interrupt priority level
(e) interrupt vector table
(f) name of the process that was executing
(g) elapsed time since process last started executing

4.13. Page reference and page-dirty bits are often implemented in software
(a) to simplify the hardware
(b) to make it possible to page the page table
(c) because the interpretation of "referenced" and "dirty" is application-specific
(d) because only the kernel has access to the clock

4.14. Thrashing may occur in a program exhibiting a high degree of temporal locality. T / F.

4.15. Larger programs are less likely to exhibit either spatial or temporal locality. T / F.

4.16. Consider a modern desktop computer on which the hard disk is spinning. The more significant delay in reading from a 4K byte file that has not been accessed in a long time is:
(a) context switch to enter the operating system
(b) time spent in the OS to determine what disk blocks to fetch
(c) latency awaiting disk arm movement
(d) latency awaiting disk rotation
(e) transfer time

4.17. In UNIX, the reference count in the inode for a file equals the sum of the number of hard links to that file, and the number of symbolic links to it. T / F.

4.18. When applications make use of shared memory, even if the shared memory region appears at different addresses in the different address spaces, there is only one set of underlying physical pages. T / F.

4.19 A micro-kernel is a category of operating systems designed for sensors or other forms of micro-processors. T / F.

4.20. An advantage of the UDP protocol relative to the TCP protocol is that:
(a) The user can send large amounts of data in a single operation
(b) Data that isn’t lost in the network will often reach the destination faster
(c) UDP, but not TCP, can be used in systems with firewalls or network address translators
(d) UDP has better security properties than TCP

4.21. A priority inversion is said to occur if a higher priority task is waiting for a lower-priority task to perform some action. T / F.

4.22. An ACL (access control list) lists all the resources that a given user has permission to access and for each resource specifies that user’s access rights. T / F.

4.23. The Unix file system scatters blocks of a file. However, when an application reads a large file sequentially, the file system prefetches blocks, and the disk device driver reorders those requests to minimize seek time, masking what could otherwise be a major inefficiency. T / F.
Question 5. 15 points.

For this question, assume a “classical” version of TCP, where during slow start, the congestion window is doubled every Round Trip Time (RTT):

\[ \text{cwnd} \leftarrow 2 \times \text{cwnd} \]

During the additive-increase-multiplicative-decrease (AIMD), the congestion window is incremented each RTT if no congestion is sensed:

\[ \text{cwnd} \leftarrow \text{cwnd} + 1 \]

And during AIMD, when congestion is sensed (i.e. a packet is dropped), the congestion window is cut in half:

\[ \text{cwnd} \leftarrow \text{cwnd} / 2 \]

Though real TCP behavior is more complex than this, assume for the purpose of this question that slow start occurs exactly once, at the beginning of the connection, followed throughout by AIMD. Assume also that congestion is detected immediately when the bottleneck bandwidth is reached. For parts 5.1 and 5.2 of this question, assume no queues between source and destination, and that the source senses congestion as soon as the throughput reaches the available bandwidth. Assume no packet loses other than those due to congestion.

For all three parts, assume an RTT of 100ms for each connection (that is, the latency with no buffering), and an initial cwnd of one packet. Assume that all data packets (i.e. packets with a TCP payload) are the same size. For all three parts, state any other assumptions that affected your answers.

5.1 [2.5 points]: Consider two independent TCP connections, where one connection has 200 packets-per-second (pps) of available bandwidth, and where the other has 2 Mpps (Mega-pps) of available bandwidth. Each connection has 20 packets to transfer (i.e. a file that can fit within 20 packets). Which connection will complete its transfer first, and why?

5.2 [2.5 points]: Now consider the case where each connection is transferring an infinitely long file. At steady state (i.e. in AIMD mode), what is the approximate average throughput achieved by each connection, and why? What is the approximate average one-way latency (where latency is defined as the time it takes a packet to get from source to destination), and why?
5.3: **[10 points]** Explain, briefly but clearly, how TCP behaves when running over a network consisting of multiple links and routers (i.e., with competing traffic and finite buffers). Assume that the sender is transmitting a very large file and hence that the sender-side TCP connection always has data to send. For full credit, you should also describe what effect the scenarios shown below have on the connections as compared with those of question 5.2. For instance, broadly speaking, what happens to the steady-state throughput and latency?

![Diagram of network with two sources and two destinations through routers and links with different capacities.](image-url)
Question 6. 15 points.

6.1[5 points] Consider a processor on which memory addresses can be split into two fields: an index into the cache, and a tag to check whether the data at that index is the data requested. Assuming we have 32-bit addresses and an 8KB direct-mapped cache, organized into 8-byte blocks, what are the total number of blocks and the total number of tag bits for a direct mapped organization? What are the number of blocks and total number of tag bits in an address for two-way and four-way organizations?
6.2 [10 points] Execution ordering on a modern processor.

6.2a. [4 points] What is out-of-order execution, and why do modern general-purpose processors use it? Be sure to define the term and describe its advantages.

6.2b. [2 points] Name two disadvantages of adding support for out-of-order execution to a general-purpose processor. Very briefly, justify your answer.

6.2c. [2 points] What is speculative execution, and how does it differ from out-of-order execution? Why do most processors that support speculative execution also support out-of-order execution?

6.2d. [2 points] Define the term "precise exception," and describe its relationship, if any, to out-of-order execution.