1. The Alternative Facts Journal (AFJ) has become one of the international's largest sources of news and current information. On a typical day, news stories (entirely new, or updated versions of existing stories) come out at an average rate of once every 15 minutes. AFJ’s popularity, international distribution span, and (mostly) static content make it a perfect use case for a CDN. Your CDN startup is trying to get AFJ’s business.

   a. You are trying to convince AFJ management of the value of using a CDN. What two advantages of using a CDN (from AFJ’s viewpoint) would you stress? A skeptic of CDNs on the AFJ management wants to derail your efforts. What is one potential shortcoming of a CDN that he could point out?

   b. Once you succeed in winning AFJ’s business, you have to design a cache consistency mechanism for their expected workload. You have 40 CDN sites, each of which receives an average number of 300 requests per second to fetch some news article. You first consider using check-on-use as the cache consistency mechanism - in other words, before handing out a cache copy of an article to a user, the CDN site checks with AFJ to make sure that the cache copy is up to date. Suppose it costs the AFJ IT infrastructure 0.01 cent to handle such a check-on-use request. What is the minimum annual IT budget of AFJ?

   c. Terrified by the answer to b., AFJ’s management begs you to reduce their IT expense. You therefore decide to use a callback-based mechanism in which a callback break message from AFJ pushes the bits of new articles (or the new version of an existing article). You negotiate a payment by AFJ of one cent per interaction with a CDN site. What is the new minimum annual IT budget of AFJ?

2. Impressed by your performance in 15-440 at CMUQ, DropBox hires you to replace their poorly-performing full-replica design with a new product that uses a genuine caching mechanism. This new product is targeted at a group of companies in the movie animation industry whose workloads are video reviewing (read-only) & video editing (read-write). Video files range from 50MB to 150MB in size, with a mean of 100MB. About 90% of the opens are for video reviewing, and about 10% are for video editing. A typical review session takes 30 min, while a typical editing session only takes 10 min. The companies that will use this system are globally distributed across the Internet, with a mean end-to-end bandwidth of 10 Mbps and a mean RTT of 100ms.

   For the new DropBox product, you architect a write-through whole-file caching system. In particular, on a close after a video editing session, the entire contents of the just-edited file are transmitted to the server. You use read & write leases on whole files as the cache consistency mechanism.

   a. Suppose you would like most sessions to complete with a single lease request. In other words, you would like to reduce the number of lease renewal requests. How long should read leases and write leases be in order to achieve this goal? State and justify any assumptions that you make.

   b. Your DropBox team has implemented & deployed this system, using the lease periods stated in your answer to a. Now consider a situation where a new lease request arrives at the server when there are 5 other requests already waiting ahead of it in a FIFO queue for a lease. What is the worst case waiting time for this new request, before it receives its lease?

   c. For b., what is the best case waiting time?
3. An Internet service makes available non-sensitive, anonymous user data that can be used for data mining purposes by customers. This data is stored in a read-only in-memory database consisting of one million equal-sized blocks of 32KB each. The blocks map to the following non-overlapping sets: 500 descriptor blocks, 1,000 tag blocks, and 5,000 index blocks, with everything else being data blocks. Database code executing on clients can directly access these blocks over the Internet.

A client implements a cache of size 10,000 blocks. Accessing a block takes 5 ms on a cache hit, and 200 ms on a cache miss. In a typical database operation at the client, 5% of the accesses are uniformly distributed over the descriptor blocks, 5% of the accesses are uniformly distributed over the tag blocks, 10% of the accesses are uniformly distributed over the index blocks and the remaining 80% of the accesses are uniformly distributed over the data blocks.

For the following questions, you may need to make simplifying assumptions. Be sure to clearly state them, and explain why they are reasonable assumptions.

a. How would you best use the available cache space? *(Hint: You may want to combine static and dynamic policies.)*

b. The cache starts out completely cold. What is the point at which you would consider the cache fully warmed up? *(Hint: A precise qualitative answer is acceptable.)*

c. Once the cache has been warmed up (as defined in your answer to b.), what is the average time for a typical client access?

*Handout continues on the next page(s)*
A server stores key-value pairs, where the value is an integer and the key is a unique identifier provided by the client which supplies that value. There are two RPC operations:

- **write(key k, int v)** changes the data object identified by key k to have new value v. If key k does not already exist, it creates a new object with value v.
- **read(key k)** returns the value of the object identified by key k if it exists.

Each client can go through one of 3 caching proxies: A, B, and C. For example, calling **B.write(K1, 4)** means write via proxy B. Assume each proxy has a LRU cache that is able to fit exactly 3 entries. For the following time sequence of operations, answer each of the questions below. Show your working.

<table>
<thead>
<tr>
<th>Time</th>
<th>Topic Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>A.write(K1, 1)</td>
</tr>
<tr>
<td>02</td>
<td>B.write(K2, 1)</td>
</tr>
<tr>
<td>03</td>
<td>C.read(K1)</td>
</tr>
<tr>
<td>04</td>
<td>B.write(K1, 0)</td>
</tr>
<tr>
<td>05</td>
<td>A.write(K1, 1)</td>
</tr>
<tr>
<td>06</td>
<td>A.write(K2, 4)</td>
</tr>
<tr>
<td>07</td>
<td>A.write(K3, 2)</td>
</tr>
<tr>
<td>08</td>
<td>C.write(K3, 7)</td>
</tr>
<tr>
<td>09</td>
<td>B.read(K1)</td>
</tr>
<tr>
<td>10</td>
<td>C.write(K4, 10)</td>
</tr>
<tr>
<td>11</td>
<td>B.read(K3)</td>
</tr>
<tr>
<td>12</td>
<td>C.write(K2, 0)</td>
</tr>
<tr>
<td>13</td>
<td>A.read(K4)</td>
</tr>
<tr>
<td>14</td>
<td>A.write(K5, 2)</td>
</tr>
<tr>
<td>15</td>
<td>C.read(K2)</td>
</tr>
</tbody>
</table>

a. Suppose a callback-with-new-value mechanism for cache consistency is used (i.e., invalidation plus new value is provided by a callback break). What does the cache of A, B and C look like after \( t = 4 \)? What about after \( t = 12 \)?

b. Instead of callback, suppose the cache uses "check on use"? What will the contents of A, B, and C’s cache be after \( t = 4 \)? What about after \( t = 14 \)?

c. Suppose faith-based caching is used, with cache entries being assumed valid for 10 time units without checking. When is the earliest that one-copy semantics is violated in the above operation sequence?

*Handout continues on the next page(s)*
5. Consider a Linux uniprocessor system that is managing an I/O buffer cache of 100 4KB blocks using the ARC (Adaptive Replacement Cache) algorithm. The total storage size is 4 GB, so there are $2^{30}$ total blocks (roughly one million) in the system. Answer the questions below by drawing the state of the ARC cache, using the notation from class. We expect to see lists $T_1$, $B_1$, $T_2$ and $B_2$ clearly identified in your illustrations, and their contents labeled if known.

(Hint: using the notation from class, the quantity $c$ is 100 here.)

a. Initially, there is just a single process $P_1$ in the system. It executes in a tight loop whose working set is just 8 blocks. For simplicity, you can assume that these are blocks 0, 1, 2, ... 7. Show the state of the ARC cache after one million iterations of the loop.

b. Suppose a second process $P_2$ is launched at the same time as $P_1$. Starting at block 100, this process sequentially scans all the blocks up to 100,000. The interleaving pattern of $P_1$ and $P_2$ will clearly influence the cache contents. Suppose Murphy (an evil background process) influences the scheduling of $P_1$ and $P_2$ to make the cache state as bad as possible for $P_1$. Under these conditions, draw the state of the ARC cache after $P_1$ has completed 10 iterations.

c. For question B, suppose $P_2$ was launched after $P_1$ completes one million iterations. $P_2$’s access pattern is the same as before, and Murphy tries just as hard to make the cache state as bad as possible for $P_1$. Under these conditions, draw the state of the ARC cache after $P_1$ has completed 10 iterations beyond the launch of $P_2$.

6. Consider a LRU page cache that receives the following reference stream (the page number of each reference is shown):

$$27, 12, 15, 15, 15, 27, 34, 15, 12, 27, 34, 15, 15, 15, 12, 8$$

Suppose the cache size is 3 pages, and the cache is initially empty.

a. How many misses does the cache experience for this reference stream? Show your working.

b. Which are the pages left in the cache at the end of this reference stream?

c. If an optimal replacement policy (OPT) is used instead of LRU, how many misses will there be and what will be the cache contents at the end? Explain your answer.

d. Give a recurring reference stream for which LRU is suboptimal with a cache size of 3 pages, but is optimal with a cache size of 4 pages.

7. Consider a file server that implements lease-based cache consistency. When the server becomes heavily loaded, would you suggest decreasing, increasing, or keeping intact the length of new leases it hands out to clients? Explain your answer and discuss the effects of this strategy on the server and clients.
8. You have been nominated as the president and official web guru of DundeeAndMe.net, an unofficial fansite paying tribute to Paul Hogan and all things Crocodile Dundee. Due to the incredible growth of the site’s popularity, you realized that having just a single web server can no longer handle the load. Hence, you decided to replicate the server, placing three replicas in different Australian cities (where the majority of the fan base resides), a single replica in southern Germany (where there is a small, but devoted cult), and a fifth replica in Boca Raton, Florida.

a. Assume you decided to use 2PC among the replicas for performing updates on site contents. List two advantages of using 2PC in this situation.

b. A freak tsunami destroyed the physical termination point of the main undersea cable connecting Australia and Asia. The Internet was effectively partitioned, separating Australia from the rest of the universe. Explain how this will affect users in Australia and in Germany, both when browsing the site as well as creating new blog entries.

c. Suppose you decided to use a Gifford-based voting scheme instead of 2PC to manage the replicas. In addition, assume a write quorum of 3 and an equal voting weight of 1 among all servers. Under the same partition conditions of part B, explain how users in Australia and in Germany will be affected, both when browsing the site as well as creating new blog entries.

d. Suppose you still wanted to employ Gifford voting with an equal weight of 1 across all servers, but now with a read quorum of 2. Again, under the same partition conditions of part B, explain how users in Australia and in Germany will be affected when browsing the site and creating new blog entries.

e. After some long daunting days, the undersea cable was finally restored, putting an end to this unpleasant network partition. During this time, however, the fortunate users who were able to continue using your site made several updates to the site’s contents, which of course were not reflected on all replicas. Assuming a Gifford-based replica management scheme like the one discussed in part C, what recovery steps (if any) should you apply in order to ensure proper functioning of the replica set once network connectivity is restored?
CNP bank uses a group of servers to coordinate and synchronize critical banking transactions. In particular, a Paxos-style protocol is employed on all the servers so as to achieve consensus on each operation. For any submitted operation, let \( I \to J \) denote a successfully delivered message from server I to server J. Recall the 4 types of messages that are used in Paxos, which can be written as follows:

- **Prepare\( (n) \)**, where \( n \) is a unique sequence (or round) number
- **Promise\( (n, (n_k, a_k)) \)**, where \( n_k \) and \( a_k \) are the last sequence number and value, respectively accepted by Acceptor \( k \) (if any) and stored at its stable storage
- **PleaseAccept\( (n, v) \)**, where \( v = a_k \) of highest \( n_k \) seen among the promise responses, or any value if no promise response contained a past accepted proposal
- **Accept_OK\()**

a. Assume 3 servers, S1, S2, and S3 are involved, and they all start out with no past accepted proposals (i.e., no sequence numbers and values are stored at their stable storages). A server can act as a Proposer, an Acceptor, or both. Suppose also that servers S1 and S2 submit proposals to quorums of Acceptors as shown in the Paxos communication trace below. What are the quorum sizes of S1 and S2, assuming that both will not submit prepare messages other than what is shown in the trace? Also, which proposal (or proposals) will achieve consensus? Explain your reasoning.

```
S1 \to S1:  Prepare(101)
S1 \to S1:  Promise(101, null)
S1 \to S2:  Prepare(101)
S1 \to S3:  Prepare(101)
S2 \to S1:  Promise(101, null)
S2 \to S3:  Prepare(102)
S3 \to S2:  Promise(102, null)
S2 \to S1:  Prepare(102)
...
```

*Handout continues on the next page(s)*
b. Assume now 5 servers, S1, S2, S3, S4, and S5 are involved, and also starting out with no past accepted proposals. As in part A, a server can act as a Proposer, an Acceptor, or both. Suppose also that servers S1 and S4 submit proposals to quorums of Acceptors as shown in the Paxos communication trace below. Which proposal (or proposals) will achieve a consensus in this case, assuming the [EVENT!] in the trace is S1 crashing? Explain your reasoning.

\[
\begin{array}{l}
S1 \rightarrow S1: \text{Prepare}(101) \\
S1 \rightarrow S1: \text{Promise}(101, \text{null}) \\
S1 \rightarrow S2: \text{Prepare}(101) \\
S2 \rightarrow S1: \text{Promise}(101, \text{null}) \\
S4 \rightarrow S4: \text{Prepare}(104) \\
S4 \rightarrow S5: \text{Prepare}(104) \\
S4 \rightarrow S4: \text{Promise}(104, \text{null}) \\
S5 \rightarrow S4: \text{Promise}(104, \text{null}) \\
S1 \rightarrow S3: \text{Prepare}(101) \\
S3 \rightarrow S1: \text{Promise}(101, \text{null}) \\
S1 \rightarrow S1: \text{PleaseAccept}(101, X) \\
S1 \rightarrow S1: \text{Accept_OK()} \\
S4 \rightarrow S1: \text{prepare}(104) \\
S1 \rightarrow S2: \text{PleaseAccept}(101, X) \\
S1 \rightarrow S3: \text{PleaseAccept}(101, X) \\
\text{[EVENT!]} \\
\end{array}
\]

24pts 10. Assume that the following four processes belong to the same group, with two different senders and a possible delivery order of messages under FIFO-ordered multi-casting:

\[
\begin{array}{cccc}
\text{Process P1} & \text{Process P2} & \text{Process P3} & \text{Process P4} \\
\hline
\text{Sends M1} & \text{Receives M1} & \text{Receives M3} & \text{Sends M3} \\
\text{Sends M2} & \text{Receives M3} & \text{Receives M1} & \text{Sends M4} \\
\text{Receives M2} & \text{Receives M2} & \text{Receives M2} & \text{Receives M2} \\
\text{Receives M4} & \text{Receives M4} & \text{Receives M4} & \text{Receives M4} \\
\end{array}
\]

a. How many delivery orderings are permissible with atomic multicasting?
b. Write down all the permissible delivery orderings with FIFO atomic multicasting.
c. Assume M1 causally precedes M4. Write down all the permissible delivery orderings with causal atomic multicasting.