(1) What is “scan-resistant” about a replacement algorithm? Is LRU scan-resistant? Please explain your answer. (5 points)

**Solution:**
A *scan-resistant* replacement algorithm allows one-time accesses of blocks to pass through without polluting the cache or flushing blocks that have temporal locality out of the cache.

LRU is NOT scan-resistant, because it makes its replacement decision purely based on recency and makes one-time accessed blocks, which have small recency when they are accessed, to replace currently cached blocks.

(2) Are 2Q, LIRS, and ARC scan-resistant? For each of the replacement algorithms that is scan-resistant, indicate what type of blocks would not be replaced due to references to first-time-accessed blocks? (10 points)

**Solution:**
All of them are cache-resistant.
In 2Q, blocks in Am are protected.
In LIRS, all LIR blocks are protected.
In ARC, blocks in T2 are protected.

(3) While the kernel is aware of page accesses that occur in any processes in the system, and a process knows only the accesses of its own file pages, why does the kernel needs to consult the process about replacement decision in the design of *Application-Controlled File Caching Policy*? We know that the policy is still a global replacement policy. In contrast, in the local replacement polices the kernel determines the allotment to each process, which makes its replacement decision where only blocks in its allotment can be replaced. Briefly state why the *Application-Controlled File Caching Policy* has its performance advantages over both conventional global replacement policies and the local replacement policies? (10 points)

**Solution:**
This is because a process may have more accurate knowledge on the future access patterns of its blocks, which are helpful to make more effective replacement decision.

Compared with the conventional global replacement polices, Application-Controlled File Caching Policy can take individual process’s inputs into its replacement decision, as mentioned above. Compared with local replacement polices, it allows more flexible cache space allocation among processes to make cache space utilized more efficiently.
Regarding the summary cache, what are false misses and false hits? How does each of the mistakes affects the total cache hit ratio or the inter-proxy traffic? Why do these mistakes not affect the correctness of the caching scheme? Describe a scenario which can lead to false missies and another one that can lead to false hits? (10 points)

Solution:
False misses: the document requested is cached at some other proxy but its summary does not reflect the fact.
False hits: the document requested is not cached at some other proxy but its summary indicates that it is.

False miss reduces the hit ratio and false hit increases the inter-proxy traffic.

These mistakes would not affect correctness because the requested documents would be retrieved from the server anyway.

Because there is a delaying in updating summaries among proxies, the actual cached contents of a proxy may not be accurately reflected in its summary copies that are stored in other proxies. If a document that is recently loaded is not reflected, a false miss can occur. If a document that was reflected in the summary is replaced, then a false hit occur.

Is this statement correct and why?
“The objective of Web prefetching is to reduce access latency and Internet traffic by removing data transferring time on the critical path of processing users’ requests.” (10 points)

Solution:
This statement is NOT correct, because Web prefetching cannot reduce Internet traffic.

In a continuous Media Server that services two classes of streaming requests, Classes A and B. Streams of Class A consume data at a speed of 2 blocks/second, and streams of Class B consume data at a speed of 1 block/second. Suppose we have a disk whose bandwidth for servicing random accesses is 8 blocks/second. We have 6 streams of requests: Sk (k = 1, 2, …6). Among them S1 and S2 belong to Class A, and the rest belong to Class B. Determine if the following two rounds of stream scheduling is legitimate (causing no jitters) and why. (10 points)

Round 1: S1, S2, S3, S4, S5, S6, S1, S2 (The streams at the left side are scheduled first)
Round 2: S3, S4, S5, S6, S1, S2, S1, S2

Solution:
The scheduling can cause jitters. As an example, suppose the first block of S1 is serviced at time 0 in the Round 1 scheduling. Because S1 consumes two blocks in a second, the disk has to come back to service the stream within 0.5 second to avoid a jitter. However, the disk would be back to service the second block of S1 on time 0.75 (6/8) second, which misses the deadline of S1.
(7) In the Web caching, there are more factors involved in the replacement decision than in the block buffer caching in OS, including document size, locality, and the cost to retrieve it. Could you describe how each of these factors would affect the replacement decision? (10 points)

Solution:
Generally, a Web caching replacement algorithm prefers to cache documents of small sizes, strong locality, and high miss penalty (the cost for retrieval).

(8) There is a hybrid approach called leases to enforce (strong) consistency in Web caching. Could you briefly describe the approach? To use leases efficiently, we may selectively choose documents, in the proxy cache, on which leases apply. Write a simple policy using pseudo code that checks certain attributes of a document and decides if a lease should be signed/renewed for the document for efficient use of leases? This pseudo code will run at the proxy. (10 points)

Solution:
In the approach called lease, the server and proxy agree that the server will notify the proxy if the leased document is updated during the lease period. If the lease has not expired, the proxy can immediately service client requests from its cache. Otherwise, it must validate the cache document.

While we know lease is most effective in applying on document that is infrequently updated at the server but frequently requested from clients, we can design a policy that monitors updating frequency and requesting frequency for each cached document and selectively apply/renew lease on a basis of individual document.

timestamp(d) = the last modification time of document d, an attribute of the document
recorded_timestamp(d) = the most recent modification time, of document d, that is recorded by system.
MON_WIN_SIZE = the size of a window in which the frequencies are collected and compared.

Initialization:

\[
\text{start_time}(d) = \text{current\_system\_time}; \\
\text{lease\_on}(d) = 0; /* indicate if a lease should be applied on document d */
\]

When request for document d is received at the proxy.
d is immediately serviced if lease is on or validated/retrieved/delivered if lease is off

\[
\text{if (current\_system\_time – start_time}(d) < \text{MON\_WIN\_SIZE}) \{ \\
\quad \text{if (recorded\_timestamp}(d) < \text{timestamp}(d)) \{ /* d has been updated at server */ \\
\qquad \text{recorded\_timestamp}(d) = \text{timestamp}(d); \\
\qquad \text{lease\_on}(d)--; \\
\quad \} \\
\quad \text{else} /* d hasn’t been updated at server */ \\
\qquad \text{lease\_on}(d)++; \\
\}\]

else { /* a window-sized monitoring ends and take action about lease */
    start_time(d) = current_system_time; /* reset a new window */
    if (lease_on(d) >= 0)
        apply/renew lease for d
    else
        ; /* don’t apply/renew lease */
    lease_on(d) = 0;
}

(9) In essence, all replacement algorithms are designed to identify hot blocks according to their respective criteria so that these blocks can be provided with certain privilege of staying in the cache for longer time. For example, the criterion for LRU is that recency is 0 (once a block is accessed, it has a recency of 0), and the criterion for ARC is that the reuse distance is not greater than cache size (a block is promoted to queue L2 if it is accessed while it is in L1). Please list the criteria for 2Q and LIRS. (10 points)

Solution:
The criterion for 2Q is that the requested block is in A1 (A1in or A1out), or its reuse distance is not greater than size of A1.
The criterion for LIRS is that the requested block is in the LIRS Stack, or its reuse distance is not greater than the maximum recency of the LIR blocks.

(10) Run the LIRS algorithm on the reference sequence: E E D C B A A B C D E E D C B A, in which (1) each letter represents a reference to the corresponding block;
(2) the left-most letter represents the first reference.
The cache size is 4 blocks, among them 3 blocks are for LIR blocks (Llir = 3) and the rest of the cache blocks are used for HIR blocks (Lhir = 1).
Given the contents of stacks of the LIRS replacement algorithm right before the reference sequence, please draw the contents of stacks after each reference in the order as they appear in the sequence(using the given notations to denote blocks of different statues), indicating miss/hit for each reference and calculating hit ratio of the whole sequence.

If the hit ratio is smaller than that for sequence of the looping pattern (e.g., E A B C D E A B C D E), which is around 80%, could you give a brief explanation on their differences (Hint: consider the assumption taken by LIRS). (15 points)
The hit ratio is $9/16 = 56.3\%$, which is less than the hit ratio for the looping pattern. The reason for the lower hit ratio is that the assumption taken the LIRS algorithm is violated in the reference sequence. The assumption is that if the IRR (reuse distance) of a block is small, then its next IRR would also be small. For blocks at the beginning or end of each repeated subsequence such as A, B, and E, their reuse distance varies dramatically.

(11) This question is used to verify that students completed their lab3 independently. If you satisfactorily complete the coding, you can earn up to 5 bonus points. However, if you cannot do it or make mistakes that are utterly different from your Lab3 code, your Lab3 score will be adjusted. [DO NOT REFER TO YOUR LAB3 CODE IF YOU DO BRING IT INTO THE CLASSROOM. OTHERWISE, YOU FAIL THE WHOLE COURSE.]

Implement the following sys-call as described. The description is somewhat different from that for your Lab3.

```c
int g_self_ra_mark, g_expand_ra_mark

asmlinkage void sys_lab_set_watermarks(int *p_self_ra_mark, int *p_expand_ra_mark)
```

Here are details you must take care of in your coding (Note that the following description doesn't represent the pseudo code of the implementation. You need to figure out your own control flow)

- If a parameter is NULL, do not change its corresponding global watermark.
If a parameter is not NULL, store the original value of its corresponding watermark into the address represented by the parameter.

If the value pointed to by a parameter is between 1 and REGION_SIZE, inclusively, store the value to its corresponding watermark.

```c
int g_self_ra_mark, g_expand_ra_mark

asmlinkage void sys_lab_set_watermarks(int *p_self_ra_mark, int *p_expand_ra_mark)
{   
    int tmp;

    if (p_self_ra_mark != NULL) {
        copy_from_user(&tmp, p_self_ra_mark, sizeof(int));
        copy_to_user(p_self_ra_mar, &g_self_ra_mark, sizeof(int));
        if (tmp >= 1 && tmp <= REGION_SIZE)
            g_self_ra_mark = tmp;
    }

    if (p_expand_ra_mark != NULL) {
        copy_from_user(&tmp, p_expand_ra_mark, sizeof(int));
        copy_to_user(p_ expand_ra_mar, & g_expand_ra_mark, sizeof(int));
        if (tmp >= 1 && tmp <= REGION_SIZE)
            g_expand_ra_mark = tmp;
    }
}
```