Access transparency: Local and remote resources are accessed using the same operations. One example is the copy (cp) command. If we were to use different commands to copy files on the same machine as opposed to from two different machines, this would violate access transparency.

Location transparency: The ability to access a resource with knowledge of their location. Access to mounted NFS directories in an example.

Concurrency transparency: Even when several processes are operating on the same set of shared resources, it should appear that there is only one process operating (that is, there is no interference between processes). An example is a database being updated by multiple users.

Replication transparency: If there are multiple copies of a resource, it should still appear as if there is only one. An example is, again, updating multiple copies of a database.

Failure transparency: System failures should be concealed from users. An example is replicated servers, where a backup server takes over in the event that the primary server fails, without any intervention from the user.

(b) When developing distributed system applications, you need to use decentralized algorithms to improve scalability. Give the four characteristics of decentralized algorithms. (5 pts)

1. No machine has complete information about the system state.
2. Machine makes decisions based only on the local information.
3. Failure of one machine does not ruin the algorithm
4. There is no implicit assumption that a global clock exists.
Question 2 – Distributed Name Servers (20 pts)

(a) I am on my machine blum.sabanciuniv.edu and I would like to access URL http://www.dcs.cs.whoknowsu.ac.tw. Assume my local name server is the one covering Sabanci University domain. Also assume that tw, ac, whoknowsu and cs are administrative zones for the URL I am trying to reach. Of course, there is a root DNS server somewhere. Show, schematically, how my name can be resolved using recursive name resolution indicating, at each step, the part of the name that is resolved. Assume that no caching is applied. (10 pts)

Solution: See the figure below.
(b) Iterative and recursive name resolution in a distributed naming system have different scalability properties. Explain the advantages and disadvantages of two schemes. (5 points)

The most important difference is that iterative resolution generally incurs higher communication cost, as remote servers need to be queried from a fixed location. In contrast, recursive resolution eventually passes requests to a nearby server. In addition, recursive resolution generally allows for more effective caching of name-to-address bindings. Read the pages 198-201 of the textbook.

(c) How does caching help a name services availability? (5 pts)

Caches in naming servers and clients are used to store the address of the named entities. Even if a name server is not active, a name can still be resolved by taking advantage of the cached addresses. For example in (a) of this question the name 
http://www.dcs.cs.whoknowsu.ac.tw can still be resolved even if the name server for whoknowsu is down when the name server for ac cached the address of the name server for cs.
Question 3 (20 points) - Method Invocation

(a) Discuss differences between static and dynamic remote method invocation. List advantages and disadvantages of each. If, for example, there was a remote method called `append` that appends integer `int` to a file object `fobject`, how would the static invocation look like and how would the dynamic invocation look like (in pseudo-code: one line for each). (5 points).

In static RMI, the server objects' interfaces are known to the client application. The client code contains an explicit invocation of a particular method of the specific object that it wants to invoke. This code is compiled and at runtime, the system knows exactly which method is being invoked. If the server object's interface changes, then the client code has to be re-compiled. Since there is no runtime resolution of the interface reference, it is faster.

In dynamic RMI, the method invocation is “composed” at runtime. The client application selects at runtime which method it will invoke at a remote object. Thus, the client code contains the invocation of a system method called `invoke` whose parameters are the object that is being referenced, the method to be invoked and the input and output parameters to the method. For the example that is given, the static invocation would take the form

```java
fobject.append(int)
```

whereas the dynamic invocation would take the form

```java
invoke(fobject, id(method), int)
```

where `id(method)` returns an identifier for the method which has been previously bound to append.

(b) What does an object adapter do? (5 points)

It accepts incoming invocation requests for the objects residing under its "regime." This means that the adapter decides how the invocation is to be carried out, for example, by using a separate thread. It is also responsible for deciding whether its objects are persistent or transient, making objects available to the outside world by providing (part of) a reference, and so on. In other words, an object adapter provides a wrapper around data and operations on that data that coincides with what you would expect from an object.
(c) A client makes remote procedure calls to a server. The client takes 5 milliseconds to compute the arguments for each request, and the server takes 10 milliseconds to process each request. The local OS processing time for each send or receive operation is 0.5 milliseconds, and the network time to transmit each request or reply message is 3 milliseconds. Marshalling or unmarshalling takes 0.5 milliseconds per message.

Calculate the time taken by the client to generate and return from two requests:
(i) if it is single-threaded, and
(ii) if it has two threads that can make requests concurrently on a single processor.
You can ignore context-switching times and assume that the server is single-threaded. (10 pts)

i) time per call =
calculate arguments + marshal arguments + OS send time + message transmission + OS receive time + unmarshall arguments + execute server procedure + marshall results + OS send time + message transmission + OS receive time + unmarshal arguments
= (5 + 0.5 + 0.5 + 3 + 0.5 + 0.5 + 10 + 0.5 + 0.5 + 3 + 0.5 + 0.5)
= 5 + 8*0.5 + 2*3 + 10 ms = 5 + 4 + 6 + 10 = 25 ms.

Time for two calls = 50 ms.

ii) threaded calls:
Client does the following
1st call:
calculate arguments + marshal arguments + OS send time (call 1) = 5 + 0.5 + 0.5 = 6
2nd call:
calculate arguments + marshal arguments + OS send time (call 1) = 5 + 0.5 + 0.5 = 6
Total time = 12 ms

then client waits for reply from first call

Server gets first call after message transmission + OS receive time + unmarshal arguments = 6 + 3 + 0.5 + 0.5 = 10 ms, takes 10 + 1 to execute, marshal, send at 21 ms

server receives 2nd call before this, but works on it after 21 ms taking 10 + 1, sends it at 32 ms from start

client receives it 3 + 1 = 4 ms later i.e. at 36 ms (message transmission + OS receive time + unmarshal arguments) later

Time for 2 calls = 36 ms.
Question 4 - Locating Entities & Garbage Collection (15 pts)

a) Why is the weighted reference counting more efficient than simple reference counting. Assume communication is reliable. (5 pts)

Creation or removal of reference can be done with only a single message, as is also the case with simple reference counting. Passing a reference is much cheaper, as it can be done with only one message containing the copied reference with its partial weight set to half of the weight of the original reference. There is thus no need to contact the object, in contrast to simple reference counting.

b) In a hierarchical location service with a depth of k, how many location records need to be updated at most when a mobile entity changes its location (hint: think of deleting the previous location records first and then inserting new ones)? (5 pts)

Changing location can be described as the combination of an insert and a delete operation. An insert operation requires that at worst k+1 location records are to be changed. Likewise, a delete operation also requires changing k+1 records, where the record in the root is shared between the two operations. This leads to a total of 2k+1 records.

c) In reference listing the object skeleton checks whether each listed process is still alive by regularly sending a ping message to each of them. If no response is received after sending a ping message to process P, the process is removed from the object’s reference list. Is it always correct to remove the process? (5 pts)

No. It may be possible that the process is temporarily unreachable due to a network partition, for example, caused by a failing router or gateway. In that case, the reference is lost and the skeleton may falsely decide that the object can be removed if the list becomes empty.
Question 5 – Synchronization (30 pts)

(a) In the following figure, use Lamport logical clock rules to set timestamp values for the events that are represented by circles. Assume that all clocks are initially set to 0. (5 pts)

(b) In the diagram in part (a) of this question, add (1) a consistent cut, and (2) an inconsistent cut. Make sure you clearly label which is which. (5 pts)
(c) Look at events “c” and “j”. Can you conclude whether they are concurrent or one
happens before the other using Lamport timestamps? Explain briefly. What does
this say about Lamport timestamps? (5 pts)

The two events “c” and j are concurrent even though C(c) < C(j) since neither P2 has
any information about the event “c” nor has P1 about “j” since they haven’t
exchanged messages after these two events happened.

This tells that C(a) < C(b) does not imply a happened before b.

(d) A scheme for implementing at-most-once reliable message delivery uses
synchronized clocks to reject duplicate messages. Processes place their local clock
value (a timestamp) in the messages they send. Each receiver keeps a table giving,
for each sending process, the largest message timestamp it has seen. Assume that
clocks are synchronized to within 100 ms, and that messages can arrive at most 50
ms after transmission (15 pts)

(i) When may a process ignore a message bearing a timestamp T, if it has recorded
the last message received from that process as having timestamp T’? (5 pts)

If T ≤ T’ then the message must be a repeat.

(ii) When may a receiver remove a timestamp 175,000 (ms) from its table? (Hint:
use the receiver's local clock value.) (5pts)

The earliest message timestamp that could still arrive when the receiver’s clock
is r is r - 100 - 50. If this is to be at least 175,000 (so that we cannot mistakenly
receive a duplicate), we need r -150 = 175,000, i.e. r = 175,150.

(iii) Should the clocks be internally synchronized or externally synchronized? (5
pts)

Internal synchronization will suffice, since only time differences are relevant.

8