EXHIBIT B

Exhibit B

	Term	Defendants' Construction	Supporting Evidence
1.	linked list to store and provide records/linked list of records [Claims 1, 3, 5, and 7]	two or more records in which each record contains a pointer to the next record in the list or information indicating that there is no next record	**Intrinsic Evidence** Title: "METHODS AND APPARATUS FOR INFORMATION STORAGE AND RETRIEVAL USING A HASHING TECHNIQUE WITH EXTERNAL CHAINING AND ON-THE-FLY REMOVAL OF EXPIRED DATA." Abstract: "A method and apparatus for performing storage and retrieval in an information storage system is disclosed that uses the hashing technique with the external chaining method for collision resolution." Col. 1, ll. 58-67 and Col. 2, ll. 1-6: "Another method for resolving collisions is called "external chaining." In this technique, each hash table location is a pointer to the head of a linked list of records, all of whose keys translate under the hashing function to that very hash table address. The linked list is itself searched sequentially when retrieving, inserting, or deleting a record. Insertion and deletion are done by adjusting pointers in the linked list. External chaining is discussed in considerable detail in the aforementioned text by D. E. Knuth, in Data Structures and Program Design, Second Edition, by R. L. Kruse, Prentice-Hall, Incorporated, Englewood Cliffs, N.J., 1987, Section 6.5, "Hashing," and Section 6.6, "Analysis of Hashing," pp. 198-215, and in Data Structures with Abstract Data Types and Pascal, by D. F. Stubbs and N. W. Webre, Brooks/Cole Publishing Company, Monterey, Calif., 1985, Section 7.4, "Hashed Implementations," pp. 310-336." Col. 5, ll. 16-25: "A common collision resolution strategy, with which the present invention is concerned, is called external chaining. Under external chaining, each hash table entry stores all of the records that collided at that location by storing not the records themselves, but instead a pointer to the head of a linked list of those same records. Such linked lists are formed by storing the records individually in dynamically allocated storage and

Exhibit B

			maintaining with each record a pointer to the location of the next record in the chain of collided records." Appendix, col. 9-10: Definitions The following formal definitions are required for specifying the insertion, retrieval, and deletion procedures. They are global to all procedures and functions shown below. 1. const table_size /* Size of hash table. */ 2. type list_element_pointer = ↑ list_element /* Pointer to elements of linked list. */ 3. type list_element = /* Each element of linked list. */ record record_contents: record_type; next: list_element_pointer /* Singly-linked list. */ end 4. var table: array [0 . . . table_size − 1] of list_element_pointer /* Hash table. */ /* Each array entry is pointer to head of list. */ Initial state of table: table[i] = nil ∀ i 0 ≤ i < table_size /* Initially empty. */ See, e.g., *Data Structures and Program Design*, Robert L. Kruse (1987), p. 20: "The idea of a *linked list* is, for every record in the list, to put a pointer into the record giving the location of the next record in the list." U.S. Pat. No. 5,287,499, Col. 5, ll. 63-68 and col. 6, ll. 1-12: "A second general technique for collision resolution is called external chaining. Under external chaining, each cell in the hash table effectively stores all of the colliding records. This is accomplished by making each table entry (each cell) consist of a pointer to the head of a linked list of records. Such linked lists are formed by storing records randomly in any available storage space, but maintaining in each record a pointer to the location of the next record in the chain. When a search key is hashed to the hash table entry, the pointer located there is used to locate the first record. If the search key does not match this record, the pointer therein contained is used to locate the second record. In this way, the 'chain' of records is traversed sequentially until the desired record is located or until the end of the chain is reached (no pointer to a next record). Deletion of records simply involves adjusting the pointers to bypass the deleted record."

Exhibit B

			Extrinsic Evidence The IEEE Standard Dictionary of Electrical and Electronics Terms (6th Ed. 1996), page 590: "A list in which each item contains a pointer to the next or preceding item in the list, making it unnecessary for the items to be physically sequential." Computer Dictionary, Microsoft Press (3rd Ed. 1997), page 285: "In programming, a list of nodes or elements of a data structure connected by pointers. A singly linked list has one pointer in each node pointing to the next node in the list."
2.	automatically expiring/expired [Claims 1, 3, 5, and 7]	becoming obsolete and no longer needed or desired in the storage system because of some external condition	**Intrinsic Evidence** Col. 2, ll. 7-21: "Some forms of information are such that individual data items, after a limited period of time, become obsolete, and their presence in the storage system is no longer needed or desired. Scheduling activities, for example, involve data that become obsolete once the scheduled event has occurred. An automatically-expiring data item, once it expires, needlessly occupies computer memory storage that could otherwise be put to use storing an unexpired item. Thus, expired items must eventually be removed to reclaim the storage for subsequent data insertions. In addition, the presence of many expired items results in needlessly long search times since the linked lists associated with external chaining will be longer than they otherwise would be. The goal is to remove these expired items to reclaim the storage and maintain fast access to the data." Col. 2, ll. 11-14: "An automatically-expiring data item, once it expires, needlessly occupies computer memory storage that could otherwise be put to use storing an unexpired item." Col. 2, ll. 14-16: "Thus, expired items must eventually be removed to reclaim

			the storage for subsequent data insertions." Col. 2, ll. 57-67 and Col. 3, ll. 1-3: "In particular, during normal data insertion or retrieval probes into the data store, the expired, obsolete records are identified and removed from the external chain linked list. Specifically, expired or obsolete records in the linked list including the record to be accessed are removed as part of the normal search procedure. This incremental garbage collection technique has the decided advantage of automatically eliminating unneeded records without requiring that the information storage system be taken off-line for such garbage collection. This is particularly important for information storage systems requiring rapid access and continuous availability to the user population." Col. 5, ll. 38-44: "In some common types of data storage systems, however, the storage system is long lived and records can become obsolete merely by the passage of time or by the occurrence of some event. If such expired, lapsed, or obsolete records are not removed from the system, they will, in time, seriously degrade the performance of the retrieval system." Col. 6, ll. 5-20: "If the end of the list has not been reached as determined by decision box 33, decision box 38 is entered to determine if the record pointed to has expired. This is determined by comparing some portion of the contents of the record to some external condition. A timestamp in the record, for example, could be compared with the current time-of-day value maintained by all computers. Alternatively, the occurrence of an event can be compared with a field identifying that event in the record. In any case, if the record has not expired, decision box 39 is entered to determine if the key in this record matches the search key. If it does, the address of the record is saved in box 40 and box 41 is entered. If the record does not match the search key, the procedure bypasses box 40 and proceeds directly to box 41. In box 41, the procedure advances forward to the next record in the linked list and the procedure returns to box 33."

Exhibit B

			’120 Patent File History, August 10, 1998 Response at page 5: “Claims 1-8 of the instant application address on-the-fly deletion of at least some records from a linked list based on automatic expiration of data, whereas ‘499 teaches automatic reorganization of records from linked list structure to sequential storage structure and vice versa to facilitate system efficiency. Nowhere does ‘499 teach deletion from the system, nor does it teach regarding automatically expiring data.” U.S. Pat. No. 5,121,495, Col. 1, ll. 57-60: "Some forms of data records have a limited lifetime after which they become obsolete. Scheduling activities, for example, involves records which become obsolete after the scheduled activity has occurred." U.S. Pat. No. 5,121,495, Col. 2, ll. 23-25: "The problem, then, is to provide speed of access of hashing techniques for large and heavily used information storage systems having expiring data..." U.S. Pat. No. 5,121,495, Col. 4, ll. 23-28: In some common types of data storage systems, data records become obsolete merely by the passage of time or by the occurrence of some event. If such expired, lapsed or obsolete records are not removed from the storage table, they will, in time, seriously degrade or contaminate the performance of the retrieval system.” U.S. Pat. No. 5,121,495, Col. 5, ll. 22-26: “If the cell tested in decision box 34 is not empty, decision box 40 is entered to determine if the record in that cell has expired. This is determined by comparing some portion of the contents of the record to some external condition.” *Data Structures and Program Design*, Robert L. Kruse (1987), p. 5 (and throughout): Kruse describes the Game of Life in which cells become dead based on the condition of cells around them. **Extrinsic Evidence**

Exhibit B

			Computer Dictionary, Microsoft Press (3rd Ed. 1997), page 184: Expire – "to stop functioning in whole or in part. Beta versions of software are often programmed to expire when a new version is released." Dictionary of Computing, (Prentice Hall's Illustrated) Jonar C. Nader (3rd Edition 1998), page 230: Expiration check – "ISO A comparison of a given date with an expiration date."
3.	identifying . . . [expired ones of the records / automatically expired ones of the records] . . . [Claims 1, 3, 5, and 7]	determining whether a record is expired by comparing some portion of the contents of the record to some external condition	**Intrinsic Evidence** Col. 6, ll. 5-20: "If the end of the list has not been reached as determined by decision box 33, decision box 38 is entered to determine if the record pointed to has expired. This is determined by comparing some portion of the contents of the record to some external condition. A timestamp in the record, for example, could be compared with the current time-of-day value maintained by all computers. Alternatively, the occurrence of an event can be compared with a field identifying that event in the record. In any case, if the record has not expired, decision box 39 is entered to determine if the key in this record matches the search key. If it does, the address of the record is saved in box 40 and box 41 is entered. If the record does not match the search key, the procedure bypasses box 40 and proceeds directly to box 41. In box 41, the procedure advances forward to the next record in the linked list and the procedure returns to box 33." U.S. Pat. No. 5,121,495, Col. 5, ll. 22-26: "If the cell tested in decision box 34 is not empty, decision box 40 is entered to determine if the record in that cell has expired. This is determined by comparing some portion of the contents of the record to some external condition." *Data Structures and Program Design*, Robert L. Kruse (1987), p. 5 (and throughout): Kruse describes the Game of Life in which cells become dead based on the condition of cells around them. **Extrinsic Evidence**

Exhibit B

			Webster's Ninth New Collegiate Dictionary (1989), page 597: Identifying – "To establish the identity of." Howard Bowman et al., Modeling Garbage Collection Algorithms using CCS and Temporal Logic (abstract), ACM (1994), page 394: "Either the programmer must explicitly allocate and de-allocate such objects or an automatic storage reclamation system, a *garbage collector*, must be employed to identify at run-time which objects may be in use now and in the future, and which objects cannot be used again."
4.	identifying . . . and removing . . . when the linked list is accessed [Claims 3 and 7]	both identification and removal of the automatically expired record(s) occurs during the same traversal of the linked list	**Intrinsic Evidence** *See* Figs. 3 and 4. Col. 2, ll. 57-67 and Col. 3, ll. 1-3: "In particular, during normal data insertion or retrieval probes into the data store, the expired, obsolete records are identified and removed from the external chain linked list. Specifically, expired or obsolete records in the linked list including the record to be accessed are removed as part of the normal search procedure. This incremental garbage collection technique has the decided advantage of automatically eliminating unneeded records without requiring that the information storage system be taken off-line for such garbage collection. This is particularly important for information storage systems requiring rapid access and continuous availability to the user population." Col. 3, ll. 4-11: "More specifically, a method for storing and retrieving information records using a linked list to store and provide access to the records, at least some of the records automatically expiring, is disclosed. The method accesses the linked list of records and identifies at least some automatically expired ones of the records. It also removes at least some

Exhibit B

			automatically expired ones of the records from the linked list when the linked list is accessed." FIG.3; Col. 5, l., 57 – Col. 6, l. 27: "Starting in box 30 of the search table procedure of FIG. 3, the search key of the record being searched for is hashed in box 31 to provide the subscript of an array element. In box 32, the hash table array location indicated by the subscript generated in box 31 is accessed to provide the pointer to the target linked list. Decision box 33 examines the pointer value to determine whether the end of the linked list has been reached. If the end has been reached, decision box 34 is entered to determine if a key match was previously found in decision box 39 (as will be described below). If so, the search is successful and returns success in box 35, followed by the procedure's termination in terminal box 37. If not, box 36 is entered where failure is returned and the procedure again terminates in box 37. If the end of the list has not been reached as determined by decision box 33, decision box 38 is entered to determine if the record pointed to has expired. This is determined by comparing some portion of the contents of the record to some external condition. A timestamp in the record, for example, could be compared with the current time-of-day value maintained by all computers. Alternatively, the occurrence of an event can be compared with a field identifying that event in the record. … If decision box 38 determines that the record under question has expired, box 42 is entered to perform the on-the-fly removal of the expired record from the linked list and the return of the storage it occupies to the system storage pool, as will be described in connection with FIG. 4." Col. 6:35-38: "It can be seen that the search table procedure of FIG. 3 operates to examine the entire linked list of records of which the searched-for record is a part, and to remove expired records, returning storage to the storage pool with each removal." Col. 6, ll. 46-53: "Though the search table procedure as shown in FIG. 3, implemented in the APPENDIX as PASCAL-like pseudocode, and described above appears in connection with an information storage and retrieval system

Exhibit B

			using the hashing technique with external chaining, its on-the-fly removal technique while traversing a linked list can be used anywhere a linked list of records with expiring data appears, even in contexts unrelated to hashing." Col. 6, ll. 56-59: "The search table procedure shown in FIG. 3, implemented as pseudocode in the APPENDIX, and described above traverses the entire linked list removing all expired records as it searches for a key match." Col. 11-12, *see* pseudocode: "HEART OF THE TECHNIQUE: Traverse entire list, deleting expired records as we search" Fig. 3 shows that the Remove function (Fig. 4) is called while the linked list is being traversed – it is in the "END OF LIST" loop 33. *See* Figs. 3-7, col. col. 5, l. 53 - col. 8, l. 60; and the pseudocode in cols. 9-14. '120 Patent File History, August 10, 1998 Response at page 3: "Although it is true that in the instant application "external chaining" and "chaining" are each equivalent to being linked, '499 does not teach or suggest on-the-fly deletion of at least some records based on automatic expiration of data, which is claimed here." '120 Patent File History, August 10, 1998 Response at pages 3-4: "Item 6 states that as to claim 5 and 7, '499 does not recite the terms "linked list," "insert," "retrieve," or "delete," but instead recites "external chaining" and "storing," and that "it would have been obvious to a person of ordinary skill in the art at the time the invention was made to use a linked list of records because a chain of records chained by an external chaining generates a linked list" (sic). The'499 patent, however, does not teach means or methods for identifying and removing "at least some expired ones of the records" from the linked list "when the linked list is accessed" (see claims 5 and 7), which is taught by the instant application and is integral to claims 5 and 7. Thus, the rejection should be withdrawn."

Exhibit B

			’120 Patent File History, August 10, 1998 Response at page 5: “Claims 1-8 of the instant application address on-the-fly deletion of at least some records from a linked list based on automatic expiration of data, whereas '499 teaches automatic reorganization of records from linked list structure to sequential storage structure and vice versa to facilitate system efficiency. Nowhere does '499 teach deletion from the system, nor does it teach regarding automatically expiring data.” ’120 Patent File History, August 10, 1998 Response at page 6: “Item 11 states that claims 1-8 are rejected under 35 U.S.C. 8 103 as being unpatentable over '499 directed to the linked lists and the step of removing, as set forth in the Double Patenting discussion, which is item 6 in the Office action. Neither '499 nor Shackelford suggest what is recited in claims 1, 3, 5, and 7, for example, means and methods for identifying and removing “at least some expired ones of the records” from the linked list “when the linked list is accessed.” ’120 Patent File History, September 22, 1998 Notice of Allowability at page 2: “The prior art does not teach or fairly suggest a method and apparatus for on-the-fly deletion of records in linked lists based on automatic expiration of data as claimed. In other words, the prior art of record does not teach or fairly suggest the means (or an equivalent step in the method claim) of “means for . . . accessing a linked list, at the same time, removing .... some of the expired ones of the records in the linked list,” as recited in lines 7-8 of claim 1. Although the prior art of record (Nemes, '495 reference) teaches the use of chains of records and the deletion of records, the Applicant, in the Response dated August 11, 1998, Paper No. 5, provided arguments as to why the chain of records as taught in the '495 reference is not the same as the linked list as claimed. The Applicant also distinguishes the claimed invention over the teachings of the '499 references, see page 3, Paper no. 5.” Robert L. Kruse, Data Structures & Program Design (2nd Ed. 1987), pages 121-124, Section 4.3.1 Sub-sections 1, 2, and 3.

Exhibit B

			Extrinsic Evidence
			
			Computer Dictionary, Microsoft Press (3rd Ed. 1997), page 135: Deallocate – "To free previously allocated memory."
			
			Webster's New World Dictionary of Computer Terms (4th Ed. 1992), page 105: Deallocation – "The release of a resource by a program when the program no longer needs it."
			
			Richard Jones and Raphael Lins, Garbage Collection (1997 reprinted in 2007), page 324: Deallocation – "the return of space to the storage manager."
			
			Stanley B. Lippman, C++ Primer (1989), page 147:
			
			[snip]
			While (pt && pt-> val == val) {
			tmp= pt->next; // pointer adjustment for removal
			delete pt; // this is the memory de-allocation.
			++cnt;
			pt = tmp;
			}
			[snip]
			
			Clifford A. Shaffer, A practical introduction to Data Structures and Algorithm Analysis (2nd Ed. 2001), page 100:
			
			Remove (elem & it) {
			If (fence->next == NULL) return false;
			It = fence->next->element;
			Link <Elem>* ltem = fence->next; //pointer adjustment

			Fence->next = ltem ->next; // pointer adjustment for removal
			If (tail==ltem) tail = fence;
			Delete ltemp; // memory disposal
			Rightcnt--;
			Return true;
			}
			
			Ellis Horowitz and Sartaj Sahni, Fundamentals of Data Structures in PASCAL (1984), page 103: Shows a PASCAL procedure for delete (i.e, remove) for linked list.
			
			Procedure delete (x,y: pointer; var fist:pointer)
			Begin
			If y = nul then first = first->link;
			Else y->link = x->link; // link adjustment
			Dispose(x); //memory disposal
			end
			
			Webster's Ninth New Collegiate Dictionary (1989), page 597: Identifying – "To establish the identity of."
			
			Howard Bowman et al., Modeling Garbage Collection Algorithms using CCS and Temporal Logic (abstract), ACM (1994), page 394: "Either the programmer must explicitly allocate and de-allocate such objects or an automatic storage reclamation system, a garbage collector, must be employed to identify at run-time which objects may be in use now and in the future, and which objects cannot be used again."
			
			Moshe Augenstein and Aaron Tenenbaum, Data Structures and pl/I Programming (1979), page 280: "An item is accessed in a linked list by traversing the list from its beginning. An array implementation allows access to the nth item in a group using a single operation, while a list implementation requires n operations."

Exhibit B

			Computer Dictionary, Microsoft Press (3rd Ed. 1997), page 138: Delete – "to eliminate text, a file, or part of a document with the intention of removing the information permanently." Webster's New World Dictionary of Computer Terms (4th Ed. 1992), page 109: Delete – "To remove or eliminate, as to erase data from a field or to eliminate a record from a file." Webster's Third New International Dictionary. Merriam-Webster, Inc. (1993), page 2122: Simultaneous – "at the same time." Webster's Third New International Dictionary. Merriam-Webster, Inc. (1993), page 2602: When – "at or during the time that." Webster's Ninth New Collegiate Dictionary at 1342 (1989), page 1342: When – "at or during the time that; while."
5.	removing at least some of the expired ones of the records from the linked list when the linked list is accessed [Claims 3 and 7]	while traversing the linked list, both adjusting the pointers in the linked list to bypass the previously identified expired records and de-allocating the memory occupied by those records	**Intrinsic Evidence** Col. 2, ll. 57-67 and Col. 3, ll. 1-3: "In particular, during normal data insertion or retrieval probes into the data store, the expired, obsolete records are identified and removed from the external chain linked list. Specifically, expired or obsolete records in the linked list including the record to be accessed are removed as part of the normal search procedure. This incremental garbage collection technique has the decided advantage of automatically eliminating unneeded records without requiring that the information storage system be taken off-line for such garbage collection. This is particularly important for information storage systems requiring rapid access and continuous availability to the user population." Col. 5, ll. 25-33: "When a search key is hashed to a hash table entry, the pointer found there is used to locate the first record. If the search key does not

			match the key found there, the pointer there is used to locate the second record. In this way, the "chain" of records is traversed sequentially until the desired record is found or until the end of the chain is reached. Deletion of records involves merely adjusting the pointers to bypass the deleted record and returning the storage it occupied to the available storage pool maintained by the system." FIG.3 and FIG. 4. Col. 5:64 – Col. 6:39: "Decision box 33 examines the pointer value to determine whether the end of the linked list has been reached. …. If the end of the list has not been reached as determined by decision box 33, decision box 38 is entered to determine if the record pointed to has expired. …. If decision box 38 determines that the record under question has expired, box 42 is entered to perform the on-the-fly removal of the expired record from the linked list and the return of the storage it occupies to the system storage pool, as will be described in connection with FIG. 4. In general, the remove procedure of box 42 (FIG. 4) operates to remove an element from the linked list by adjusting its predecessor's pointer to bypass that element. (However, if the element to be removed is the first element of the list, then there is no predecessor and the hash table array entry is adjusted instead.) On completion of procedure remove invoked from box 42, the search table procedure returns to box 33. It can be seen that the search table procedure of FIG. 3 operates to examine the entire linked list of records of which the searched-for record is a part, and to remove expired records, returning storage to the storage pool with each removal. If the storage pool is depleted and many expired records remain despite such automatic garbage collection, then the insertion of new records is inhibited (boxes 76 and 77 of FIG. 5) until a deletion is made by the delete procedure (FIG. 7) or until the search table procedure has had a chance to replenish the storage pool through its on-the-fly garbage 45 collection." Col. 6, ll. 56-59: "The search table procedure shown in FIG. 3, implemented as pseudocode in the APPENDIX, and described above traverses the entire linked list removing all expired records as it searches for a key match."

Exhibit B

			Col. 7, ll. 15-42: "In FIG. 4 there is shown a flowchart of a remove procedure that removes a record from the retrieval system, either an unexpired record through the delete procedure as will be noted in connection with FIG. 7, or an expired record through the search table procedure as noted in connection with FIG. 3. In general, this is accomplished by the invoking procedure, being either the delete procedure (FIG. 7) or the search table procedure (FIG. 3), passing to the remove procedure a pointer to a linked list element to remove, a pointer to that element's predecessor element in the same linked list, and the subscript of the hash table array location containing the pointer to the head of the linked list from which the element is to be removed. In the case that the element to be removed is the first element of the linked list, the predecessor pointer passed to the remove procedure by the invoking procedure has the NIL (sometimes called NULL, or EMPTY) value, indicating to the remove procedure that the element to be removed has no predecessor in the list. The invoking procedure expects the remove procedure, on completion, to have advanced the passed pointer that originally pointed to the now-removed element so that it points to the successor element in that linked list, or NIL if the removed element was the final element. (The search table procedure of FIG. 3, in particular, makes use of the remove procedure's advancing this passed pointer in the described way; it is made use of in that box 33 of FIG. 3 is entered directly following completion of box 42, as was described above in connection with FIG. 3.)." Col. 7, ll. 43-50: "The remove procedure causes actual removal of the designated element by adjusting the predecessor pointer so that it bypasses the element to be removed. In the case that 45 the predecessor pointer has the NIL value, the hash table array entry indicated by the passed subscript plays the role of the predecessor pointer and is adjusted the same way in its stead. Following pointer adjustments, the storage occupied by the removed element is returned to the system storage pool for future allocation." Col. 7, ll. 57-64: "If so, box 54 is entered to adjust the linked list head pointer

			in the hash table array to bypass the first element, after which the procedure continues on to box 55. If not, box 53 is entered where the predecessor pointer is adjusted to bypass the 60 element to remove, after which the procedure proceeds, once again, to box 55. Finally, in box 55 the storage occupied by the bypassed element is returned to the system storage pool and the procedure terminates in terminal box 56." (FIG. 4, described by this text is referred to as a "remove procedure") Col. 8, ll. 22-25: "If decision box 76 determines that sufficient storage can be allocated from the system storage pool for a new linked list element, then box 78 is entered where the actual memory allocation is made." Col. 8, ll. 60-64: The attached APPENDIX contains PASCAL-like pseudocode listings for all of the programmed components necessary to implement an information storage and retrieval system operating in accordance with the present invention. Col. 11-12, *see* pseudocode: "HEART OF THE TECHNIQUE: Traverse entire list, deleting expired records as we search" Col. 13-14: The "remove" procedure includes both deletion (adjusting the pointer) and dispose (de-allocating). Fig. 3 shows that the Remove function (Fig. 4) is called while the linked list is being traversed – it is in the "END OF LIST" loop 33. '120 Patent File History, August 10, 1998 Response at page 3: "Although it is true that in the instant application "external chaining" and "chaining" are each equivalent to being linked, '499 does not teach or suggest on-the-fly deletion of at least some records based on automatic expiration of data, which is claimed here." '120 Patent File History, August 10, 1998 Response at pages 3-4: "Item 6

			states that as to claim 5 and 7, '499 does not recite the terms “linked list,” “insert,” “retrieve,” or “delete,” but instead recites “external chaining” and “storing,” and that “it would have been obvious to a person of ordinary skill in the art at the time the invention was made to use a linked list of records because a chain of records chained by an external chaining generates a linked list” (sic). The'499 patent, however, does not teach means or methods for identifying and removing “at least some expired ones of the records” from the linked list “when the linked list is accessed” (see claims 5 and 7), which is taught by the instant application and is integral to claims 5 and 7. Thus, the rejection should be withdrawn.” ’120 Patent File History, August 10, 1998 Response at page 5: “Claims 1-8 of the instant application address on-the-fly deletion of at least some records from a linked list based on automatic expiration of data, whereas '499 teaches automatic reorganization of records from linked list structure to sequential storage structure and vice versa to facilitate system efficiency. Nowhere does '499 teach deletion from the system, nor does it teach regarding automatically expiring data.” ’120 Patent File History, August 10, 1998 Response at page 6: “Item 11 states that claims 1-8 are rejected under 35 U.S.C. 8 103 as being unpatentable over '499 directed to the linked lists and the step of removing, as set forth in the Double Patenting discussion, which is item 6 in the Office action. Neither '499 nor Shackelford suggest what is recited in claims 1, 3, 5, and 7, for example, means and methods for identifying and removing “at least some expired ones of the records” from the linked list “when the linked list is accessed.” ’120 Patent File History, September 22, 1998 Notice of Allowability at page 2: “The prior art does not teach or fairly suggest a method and apparatus for on-the-fly deletion of records in linked lists based on automatic expiration of data as claimed. In other words, the prior art of record does not teach or fairly suggest the means (or an equivalent step in the method claim) of “means

			for . . . accessing a linked list, at the same time, removing .... some of the expired ones of the records in the linked list," as recited in lines 7-8 of claim 1. Although the prior art of record (Nemes, '495 reference) teaches the use of chains of records and the deletion of records, the Applicant, in the Response dated August 11, 1998, Paper No. 5, provided arguments as to why the chain of records as taught in the '495 reference is not the same as the linked list as claimed. The Applicant also distinguishes the claimed invention over the teachings of the '499 references, see page 3, Paper no. 5."
			
			Robert L. Kruse, Data Structures & Program Design (2nd Ed. 1987), pages 121-124, Section 4.3.1 Sub-sections 1, 2, and 3.
			
			Extrinsic Evidence
			
			Computer Dictionary, Microsoft Press (3rd Ed. 1997), page 135: Deallocate – "To free previously allocated memory."
			
			Webster's New World Dictionary of Computer Terms (4th Ed. 1992), page 105: Deallocation – "The release of a resource by a program when the program no longer needs it."
			
			Richard Jones and Raphael Lins, Garbage Collection (1997 reprinted in 2007), page 324: Deallocation – "the return of space to the storage manager."
			
			Stanley B. Lippman, C++ Primer (1989), page 147:
			
			[snip]
			
			While (pt && pt-> val == val) {
			tmp= pt->next; // pointer adjustment for removal
			delete pt; // this is the memory de-allocation.
			++cnt;

			pt = tmp;
			}
			[snip]
			
			Clifford A. Shaffer, A practical introduction to Data Structures and Algorithm Analysis (2nd Ed. 2001), page 100:
			
			Remove (elem & it) {
			If (fence->next == NULL) return false;
			It = fence->next->element;
			Link <Elem>* ltem = fence->next; //pointer adjustment
			Fence->next = ltem ->next; // pointer adjustment for removal
			If (tail==ltem) tail = fence;
			Delete ltemp; // memory disposal
			Rightcnt--;
			Return true;
			}
			
			Ellis Horowitz and Sartaj Sahni, Fundamentals of Data Structures in PASCAL (1984), page 103: Shows a PASCAL procedure for delete (i.e, remove) for linked list.
			
			Procedure delete (x,y: pointer; var fist:pointer)
			Begin
			If y = nul then first = first->link;
			Else y->link = x->link; // link adjustment
			Dispose(x); //memory disposal
			end
			
			Moshe Augenstein and Aaron Tenenbaum, Data Structures and pl/I Programming (1979), page 280: "An item is accessed in a linked list by traversing the list from its beginning. An array implementation allows access to the nth item in a group using a single operation, while a list implementation requires n operations."

Exhibit B

			Computer Dictionary, Microsoft Press (3rd Ed. 1997), page 138: Delete – "to eliminate text, a file, or part of a document with the intention of removing the information permanently." Webster's New World Dictionary of Computer Terms (4th Ed. 1992), page 109: Delete – "To remove or eliminate, as to erase data from a field or to eliminate a record from a file." Webster's Third New International Dictionary. Merriam-Webster, Inc. (1993), page 2122: Simultaneous – "at the same time." Webster's Third New International Dictionary. Merriam-Webster, Inc. (1993), page 2602: When – "at or during the time that." Webster's Ninth New Collegiate Dictionary at 1342 (1989), page 1342: When – "at or during the time that; while."
6.	dynamically determining maximum number of expired ones of the records to remove when the linked list is accessed [Claims 4 and 8]	immediately before the linked list is traversed, determining a single number that serves as an upper limit on the number of records to remove as the linked list is traversed	**Intrinsic Evidence** Col. 6, ll. 56-67 and Col. 7, ll. 1-15: "The search table procedure shown in FIG. 3, implemented as pseudocode in the APPENDIX, and described above traverses the entire linked list removing all expired records as it searches for a key match. The procedure can be readily adapted to remove some but not all of the expired records, thereby shortening the linked list traversal time and speeding up the search at the expense of perhaps leaving some expired records in the list. For example, the procedure can be modified to terminate when a key match occurs. (PASCAL-like pseudocode for this alternate version of search table appears in the APPENDIX.) The implementor even has the prerogative of choosing among these strategies dynamically at the time search table is invoked by the caller, thus sometimes removing all expired records, at other times removing some but not all of them, and yet at

			other times choosing to remove none of them. Such a dynamic runtime decision might be based on factors such as, for example, how much memory is available in the system storage pool, general system load, time of day, the number of records currently residing in the information system, and other factors both internal and external to the information storage and retrieval system itself A person skilled in the art will appreciate that the technique of removing all expired records while searching the linked list can be expanded to include techniques whereby not necessarily all expired records are removed, and that the decision regarding if and how many records to delete can be a dynamic one." '120 Patent File History, August 10, 1998 Response, at page 4: "Item 6 states that as to claims 6 and 8,' '499 does not recite a "maximum number of records" but instead recites "threshold," and that "It would have been obvious to a person of ordinary skill in the art at the time the invention was made to group a number or records for determining the threshold and thus to predetermine the maximum number for the threshold to facilitate an efficient processing of records ...." The "maximum number of records" (in the instant application) and "threshold" (in '499) serve different purposes and are structured and determined differently. In the instant application, the number is a single quantity that serves as an upper limit on the number of records removed from the linked list whenever the linked list is accessed (see claims 6 and 8), whereas in '499 the threshold is a pair of coupled quantities, an upper threshold and a lower threshold, that serve as two-way signals indicating when the system should automatically reorganize a group of records that reside in cells of the hash table into a linked list, and vice versa (col. 6, lines 44-54 and 61-65; APPENDIX). Since neither the maximum number of records nor the upper threshold can be learned from the other by a person of ordinary skill in the art from either '499 or the instant application, the rejection should be removed." '120 Patent File History, August 10, 1998 Response, at pages 5-6: "The instant application teaches and claims (claims 2, 4, 6, and 8) means and

			method for dynamically determining the maximum number of records to be removed on-the-fly from a linked list when that linked list is accessed. Shackelford, on the other hand, teaches an unrelated quantity, the existence of a stored quantity accompanying the stream class data structure that identifies the maximum number of pointers that are permitted to exist (col. 3, line 61 through col. 4, line 2). Shackelford does not address an application with automatically expiring data, nor does he address how many items to delete. These references separately or in combination do not suggest the claims of the present application. The rejection, therefore, should be withdrawn." **Extrinsic Evidence** Webster's Ninth New Collegiate Dictionary (1989), page 734: Maximum -- "the greatest quantity or value attainable or attained." Webster's Third New International Dictionary. Merriam- Webster, Inc. (1993), page 1396: Maximum – "the greatest quantity or value attainable in a given case." Computer Dictionary, Microsoft Press (3rd Ed. 1997), page, page 165: "dynamic" – "Occurring immediately and concurrently. The term is used in describing both hardware and software; in both cases it describes some action or event that occurs when and as needed. In dynamic memory management, a program is able to negotiate with the operating system when it needs more memory." American Heritage Dictionary, Second College Edition (1985), page 388: "determine" – "To establish or ascertain definitely, as after consideration, investigation, or calculation." IBM Dictionary of Computing (10th ed. 1994), page 224: dynamic – "(1) In programming languages, pertaining to properties that can only be established during the execution of a program, for example, the length of a variable-

Exhibit B

			length data object is dynamic. (2) Pertaining to an operation that occurs at the time it is needed rather than at a predetermined or fixed time. (3) Pertaining to events occurring at run time, or during processing. (4) Contrast with static."
7.	a record search means utilizing a search key to access the linked list [Claim 1] a record search means utilizing a search key to access a linked list of records having the same hash address [Claim 5]	Means plus function limitation. Indefinite.	For all means-plus-function limitations, see that which is cited for the words/phrases within the means-plus-function limitation above. **Intrinsic Evidence** Col. 4:67 – 5:15: "A hashing function translates the key into a hash table array subscript, which is used as an index into the array where searches for the data record begin. The hashing function can be any operation on the key that results in subscripts mostly uniformly distributed across the table. Known hashing functions include truncation, folding, transposition, modulo arithmetic, and combinations of these operations. Unfortunately, hashing functions generally do not produce unique locations in the hash table, in that many distinct keys map to the same location, producing what are called collisions. Some form of collision resolution is required in all hashing systems. In every occurrence of collision, finding an alternate location for a collided record is necessary. Moreover, the alternate location must be readily reachable during future searches for the displaced record."
8.	a hashing means to provide access to records stored in a memory of the system and using an external chaining technique to store the records with same hash	Means plus function limitation. Indefinite.	For all means-plus-function limitations, see that which is cited for the words/phrases within the means-plus-function limitation above. **Intrinsic Evidence** Col. 2. 4:67 – 5:15: "A hashing function translates the key into a hash table array subscript, which is used as an index into the array where searches for the data record begin. The hashing function can be any operation on the key that results in subscripts mostly uniformly distributed across the table. Known hashing functions include truncation, folding, transposition, modulo arithmetic, and combinations of these operations. Unfortunately, hashing functions generally do not produce unique locations in the hash table, in that

Exhibit B

	address, at least some of the records automatically expiring [Claim 5]		many distinct keys map to the same location, producing what are called collisions. Some form of collision resolution is required in all hashing systems. In every occurrence of collision, finding an alternate location for a collided record is necessary. Moreover, the alternate location must be readily reachable during future searches for the displaced record."
9.	means for identifying and removing at least some of the expired ones of the records from the linked list when the linked list is accessed [Claim 1] means for identifying and removing at least some expired ones of the records from the linked list of records when the linked list is accessed [Claim 5]	Means plus function limitation. Function: identifying and removing at least some [of the] expired ones of the records from the linked list [of records] when the linked list is accessed. Both identification and removal of an automatically expired record occurs during the same traversal of the linked list. For claim 1, the phrase "when the linked list is accessed" refers to the time during which the "utilizing a search key to access the linked list" function in limitation is carried out in claim 1. For claim 5, the phrase "when the linked list is accessed"	For all means-plus-function limitations, see that which is cited for the words/phrases within the means-plus-function limitation above. **Intrinsic Evidence** *See* Claims 1 and 5; Figs. 1, 3, and 4; col. 3, l. 53 to col. 4, l. 22 and col.5, l. 53 to col. 7, l. 64; and pseudocode at cols. 11-14. Col. 5, ll. 53-57: "Referring then to FIG. 3, there is shown a flowchart of a search table procedure for searching the hash table preparatory to inserting, retrieving, or deleting a record, in accordance with the present invention, and involving the dynamic removal of expired records in a targeted linked list." Col. 6, ll. 5-34 (emphasis added): "If the end of the list has not been reached as determined by decision box 33, decision box 38 is entered to determine if the record pointed to has expired. This is determined by comparing some portion of the contents of the record to some external condition. A timestamp in the record, for example, could be compared with the current time-of-day value maintained by all computers. Alternatively, the occurrence of an event can be compared with a field identifying that event in the record. In any case, if the record has not expired, decision box 39 is entered to determine if the key in this record matches the search key. If it does, the address of the record is saved in box 40 and box 41 is entered. If the record does not match the search key, the procedure bypasses box 40 and proceeds directly to box 41. In box 41, the procedure advances forward to the next record in the linked list and the procedure returns to box 33.

		refers to the time during which the "utilizing a search key to access a linked list of records having the same hash address" function is carried out in claim 5. Removing requires, while traversing the linked list, both adjusting the pointers in the linked list to bypass the previously identified expired records and de-allocating the memory occupied by those records. Means disclosed: Boxes 10 and 11 of Fig. 1, Boxes 38 and 42 of Fig. 3, Fig 4, pseudocode in the Search Procedure (cols. 11-14) and Remove Procedure (cols. 13-14), and corresponding portions of the specification. The inclusion of "the records search means," however, renders these limitations indefinite as the "record search means" limitation is indefinite.	If decision box 38 determines that the record under question has expired, box 42 is entered to perform the on-the-fly removal of the expired record from the linked list and the return of the storage it occupies to the system storage pool, as will be described in connection with FIG. 4**.** In general, the remove procedure of box 42 (FIG. 4) operates to remove an element from the linked list by adjusting its predecessor's pointer to bypass that element. (However, if the element to be removed is the first element of the list, then there is no predecessor and the hash table array entry is adjusted instead.) On completion of procedure remove invoked from box 42, the search table procedure returns to box 33." Col. 6, ll. 35-39: "It can be seen that the search table procedure of FIG. 3 operates to examine the entire linked list of records of which the searched-for record is a part, and to remove expired records, returning storage to the storage pool with each removal. If the storage pool is depleted and many expired records remain despite such automatic garbage collection, then the insertion of new records is inhibited (boxes 76 and 77 of FIG. 5) until a deletion is made by the delete procedure (FIG. 7) or until the search table procedure has had a chance to replenish the storage pool through its on-the-fly garbage collection." Col. 6, ll. 46-55: "Though the search table procedure as shown in FIG. 3, implemented in the APPENDIX as PASCAL-like pseudocode, and described above appears in connection with an information storage and retrieval system using the hashing technique with external chaining, its on-the-fly removal technique while traversing a linked list can be used anywhere a linked list of records with expiring data appears, even in contexts unrelated to hashing. A person skilled in the art will appreciate that this technique can be readily applied to manipulate linked lists not necessarily used with hashing." Col. 6, ll. 56-67 and Col. 7, ll. 1-15: The search table procedure shown in FIG. 3, implemented as pseudocode in the APPENDIX, and described above traverses the entire linked list removing all expired records as it searches for a

			key match. The procedure can be readily adapted to remove some but not all of the expired records**,** thereby shortening the linked list traversal time and speeding up the search at the expense of perhaps leaving some expired records in the list. For example, the procedure can be modified to terminate when a key match occurs. (PASCAL-like pseudocode for this alternate version of search table appears in the APPENDIX.) The implementor even has the prerogative of choosing among these strategies dynamically at the time search table is invoked by the caller, thus sometimes removing all expired records, at other times removing some but not all of them, and yet at other times choosing to remove none of them. Such a dynamic runtime decision might be based on factors such as, for example, how much memory is available in the system storage pool, general system load, time of day, the number of records currently residing in the information system, and other factors both internal and external to the information storage and retrieval system itself A person skilled in the art will appreciate that the technique of removing all expired records while searching the linked list can be expanded to include techniques whereby not necessarily all expired records are removed, and that the decision regarding if and how many records to delete can be a dynamic one." '120 Patent File History, August 10, 1998 Response at page 3: "Although it is true that in the instant application "external chaining" and "chaining" are each equivalent to being linked, '499 does not teach or suggest on-the-fly deletion of at least some records based on automatic expiration of data, which is claimed here." '120 Patent File History, August 10, 1998 Response at pages 3-4: "Item 6 states that as to claim 5 and 7, '499 does not recite the terms "linked list," "insert," "retrieve," or "delete," but instead recites "external chaining" and "storing," and that "it would have been obvious to a person of ordinary skill in the art at the time the invention was made to use a linked list of records because a chain of records chained by an external chaining generates a linked list" (sic). The'499 patent, however, does not teach means or methods for

			identifying and removing “at least some expired ones of the records” from the linked list “when the linked list is accessed” (see claims 5 and 7), which is taught by the instant application and is integral to claims 5 and 7. Thus, the rejection should be withdrawn.” ’120 Patent File History, August 10, 1998 Response at page 5: “Claims 1-8 of the instant application address on-the-fly deletion of at least some records from a linked list based on automatic expiration of data, whereas '499 teaches automatic reorganization of records from linked list structure to sequential storage structure and vice versa to facilitate system efficiency. Nowhere does '499 teach deletion from the system, nor does it teach regarding automatically expiring data.” ’120 Patent File History, August 10, 1998 Response at page 6: “Item 11 states that claims 1-8 are rejected under 35 U.S.C. 8 103 as being unpatentable over '499 directed to the linked lists and the step of removing, as set forth in the Double Patenting discussion, which is item 6 in the Office action. Neither '499 nor Shackelford suggest what is recited in claims 1, 3, 5, and 7, for example, means and methods for identifying and removing “at least some expired ones of the records” from the linked list “when the linked list is accessed.” ’120 Patent File History, September 22, 1998 Notice of Allowability at page 2: “The prior art does not teach or fairly suggest a method and apparatus for on-the-fly deletion of records in linked lists based on automatic expiration of data as claimed. In other words, the prior art of record does not teach or fairly suggest the means (or an equivalent step in the method claim) of “means for . . . accessing a linked list, at the same time, removing .... some of the expired ones of the records in the linked list,” as recited in lines 7-8 of claim 1. Although the prior art of record (Nemes, '495 reference) teaches the use of chains of records and the deletion of records, the Applicant, in the Response dated August 11, 1998, Paper No. 5, provided arguments as to why the chain of records as taught in the '495 reference is not the same as the linked list as

			claimed. The Applicant also distinguishes the claimed invention over the teachings of the '499 references, see page 3, Paper no. 5."
10.	means, utilizing the record search means, for accessing the linked list and, at the same time, removing at least some of the expired ones of the records in the linked list [Claim 1]	Means plus function limitation. Function: using the record search means defined above, accessing and during the same traversal of the linked list removing at least some of the expired ones of the records in the linked list. "at the same time" means during the same traversal of the linked list. This limitation requires that the referenced means remove at least one of the expired ones of the records in the linked list while utilizing a search key to access the linked list. Removing requires, while traversing the linked list, both adjusting the pointers in the linked list to bypass the previously identified expired records and de-allocating the	For all means-plus-function limitations, see that which is cited for the words/phrases within the means-plus-function limitation above. **Intrinsic Evidence** *See* Claim 1; Figs. 1, 4-7; col. 3, l. 53 - col. 4, l. 22 and col. 7, l. 16 - col. 9, l. 2; and pseudocode at cols. 9-14. *See* Intrinsic Evidence citations related to "removing" under Term 9 *supra*.

		memory occupied by those records. Means: Boxes 10 and 11 of Fig. 1; Figs. 4, 5, 6, and 7, pseudocode in the Search Procedure (cols. 11-14), Insert Procedure (cols. 9 and 10), Retrieve Procedure (cols. 9 and 10), Delete Procedure (cols. 11-12), and Remove Procedure (cols. 13-14), and corresponding portions of the specification. Inserting, retrieving, and deleting are all required. The inclusion of "utilizing the records search means," however, renders this limitation indefinite as the "record search means" limitation is indefinite.	
11.	mea[n]s, utilizing the record search means, for inserting, retrieving, and deleting records from the system and, at the same time, removing	Means plus function limitation. Function: Using the record search means defined above, inserting, retrieving, and deleting during the same traversal of the linked list removing at least some expired ones of the records in	For all means-plus-function limitations, see that which is cited for the words/phrases within the means-plus-function limitation above. **Intrinsic Evidence** *See* Claim 5; Figs. 1, 4-7; col. 3, l. 53 - col. 4, l. 22 and col. 7, l. 16 - col. 9, l. 2; and pseudocode at cols. 9-14. *See* Intrinsic Evidence citations related to “removing” under Term 9 *supra*.

	at least some expired ones of the records in the accessed linked list of records [Claim 5]	the accessed linked list of records. "at the same time" means during the same traversal of the linked list. This limitation requires that the referenced means remove at least one of the expired ones of the records in the linked list while utilizing a search key to insert, retrieve, and delete records having the same hash address from the system. Removing requires, while traversing the linked list, both adjusting the pointers in the linked list to bypass the previously identified expired records and de-allocating those records from memory Means: Boxes 10 and 11 of Fig. 1; Figs. 4, 5, 6, and 7, pseudocode in the Search Procedure (cols. 11-14), Insert Procedure (cols. 9 and 10), Retrieve Procedure (cols. 9 and 10), Delete Procedure (cols. 11-12), and Remove	

Exhibit B

		Procedure (cols. 13-14), and corresponding portions of the specification. Inserting, retrieving, and deleting are all required. The inclusion of "utilizing the records search means," however, renders this limitation indefinite as the "record search means" limitation is indefinite.	
12.	means for dynamically determining maximum number for the record search means to remove in the accessed linked list of records [Claims 2 and 6]	Means plus function limitation Indefinite.	For all means-plus-function limitations, see that which is cited for the words/phrases within the means-plus-function limitation above. **Intrinsic Evidence** Col. 6, ll. 56-67 and Col. 7, ll. 1-15: "The search table procedure shown in FIG. 3, implemented as pseudocode in the APPENDIX, and described above traverses the entire linked list removing all expired records as it searches for a key match. The procedure can be readily adapted to remove some but not all of the expired records, thereby shortening the linked list traversal time and speeding up the search at the expense of perhaps leaving some expired records in the list. For example, the procedure can be modified to terminate when a key match occurs. (PASCAL-like pseudocode for this alternate version of search table appears in the APPENDIX.) The implementor even has the prerogative of choosing among these strategies dynamically at the time search table is invoked by the caller, thus sometimes removing all expired records, at other times removing some but not all of them, and yet at other times choosing to remove none of them. Such a dynamic runtime decision might be based on factors such as, for example, how much memory is available in the system storage pool, general system load, time of day, the number of records currently residing in the information system, and other

			factors both internal and external to the information storage and retrieval system itself. A person skilled in the art will appreciate that the technique of removing all expired records while searching the linked list can be expanded to include techniques whereby not necessarily all expired records are removed, and that the decision regarding if and how many records to delete can be a dynamic one." '120 Patent File History, August 10, 1998 Response, at page 4: "Item 6 states that as to claims 6 and 8,' '499 does not recite a "maximum number of records" but instead recites "threshold," and that "It would have been obvious to a person of ordinary skill in the art at the time the invention was made to group a number or records for determining the threshold and thus to predetermine the maximum number for the threshold to facilitate an efficient processing of records ...." The "maximum number of records" (in the instant application) and "threshold" (in '499) serve different purposes and are structured and determined differently. In the instant application, the number is a single quantity that serves as an upper limit on the number of records removed from the linked list whenever the linked list is accessed (see claims 6 and 8), whereas in '499 the threshold is a pair of coupled quantities, an upper threshold and a lower threshold, that serve as two-way signals indicating when the system should automatically reorganize a group of records that reside in cells of the hash table into a linked list, and vice versa (col. 6, lines 44-54 and 61-65; APPENDIX). Since neither the maximum number of records nor the upper threshold can be learned from the other by a person of ordinary skill in the art from either '499 or the instant application, the rejection should be removed." '120 Patent File History, August 10, 1998 Response, at pages 5-6: "The instant application teaches and claims (claims 2, 4, 6, and 8) means and method for dynamically determining the maximum number of records to be removed on-the-fly from a linked list when that linked list is accessed. Shackelford, on the other hand, teaches an unrelated quantity, the existence of a stored quantity accompanying the stream class data structure that identifies

Exhibit B

			the maximum number of pointers that are permitted to exist (col. 3, line 61 through col. 4, line 2). Shackelford does not address an application with automatically expiring data, nor does he address how many items to delete. These references separately or in combination do not suggest the claims of the present application. The rejection, therefore, should be withdrawn."
13.	Ordering of limitations of claim 3	The elements of claim 3 must be executed in order. Moreover, "when the linked list is accessed" in the removing step refers to the accessing step, and the identifying and removing steps must occur during the same traversal of the linked list of records.	**Intrinsic Evidence** Claim 3: "A method for storing and retrieving information records using a linked list to store and provide access to the records, at least some of the records automatically expiring, the method comprising the steps of: accessing the linked list of records, identifying at least some of the automatically expired ones of the records, and removing at least some of the automatically expired records from the linked list when the linked list is accessed." *See* Figs. 3 and 4; col. 5, l. 53 through col. 7, l. 64; and the pseudocode in cols. 11-14.
14.	Ordering of limitations of claim 7	The elements of claim 7 must be executed in order. Moreover, "when the linked list is accessed" in the removing step refers to accessing step, and the identifying and removing steps must occur during the same traversal of the linked list of records.	**Intrinsic Evidence** Claim 7: "A method for storing and retrieving information records using a hashing technique to provide access to the records and using an external chaining technique to store the records with same hash address, at least some of the records automatically expiring, the method comprising the steps of: accessing a linked list of records having same hash address, identifying at least some of the automatically expired ones of the records,

			removing at least some of the automatically expired records from the linked list when the linked list is accessed, and inserting, retrieving or deleting one of the records from the system following the step of removing." *See* Figs. 3-7; col. 5, l. 53 through col. 8, l. 60; and the pseudocode in cols. 9-14.