First Name	Last Name
First Name	Last Name

Review: LINKED LISTS

1.

(A) List one advantage of using a linked list with a sentinel nodes:

(B) List one advantage of using a linked list with two sentinel nodes:

(C) List one advantage of using a doubly-linked list:

2. One of the most common uses of doubly linked lists is:
(A) multi-linked list insert **(B)** multi-linked list delete **(C)** multi-linked list search

3.
(A) For efficiency, searching a linked list should use the binary search: **TRUE / FALSE**?

(B) Storage of files on disk: Linked allocation is essentially a disk-based version of the linked list. With linked list allocation each file is linked list of disk blocks. These disk blocks may be scattered through the disk. A few bytes of each disk block contain the address of the next block. The directory contains a pointer to the first (and last) blocks of the file.

TRUE / FALSE?

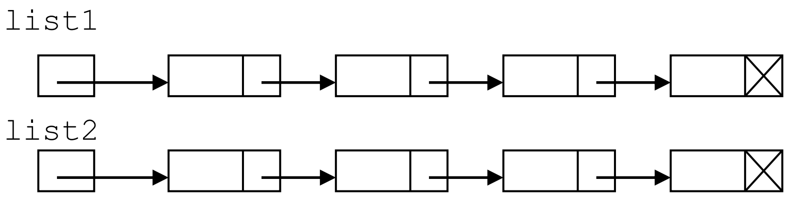
(C) One of the linked list applications is the heap storage management: **TRUE / FALSE**?

4. Searching a sorted list. Which one of the following fragments of code is incorrect? Why?

(A) `while( (*pCur)->data.key < target && *pCur != NULL )`
(B) `while( *pCur != NULL && (*pCur)->data.key < target )`

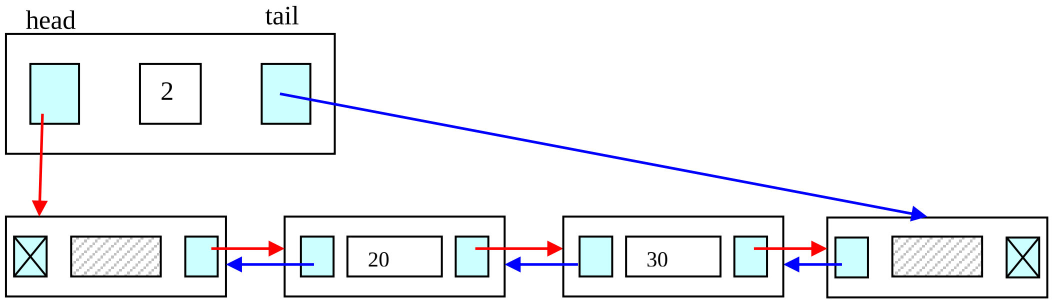
Review: LINKED LISTS

5. Imagine we have two singly linked lists as shown below. What would happen if we apply the following statements to the two lists? **Draw "the answer".**



```
pNode = NULL
pWalk = list1
loop( pWalk not NULL )
   pNode = pWalk
   pWalk = pWalk->next
end loop
pNode->next = list2->next
```

6. Write a code oriented pseudocode or a C++ function that swaps two consecutive nodes in a doubly-linked list by changing pointers (the data field inside the linked list node is not to be used); it has one parameter, a positive integer, representing the number of the first node to be swapped (for instance if the number is 2, the 2nd and the 3rd nodes are to be swapped); it returns true if possible, false otherwise. Assume the list has two sentinel nodes.



```
struct Node
{
   Node *back;
   Data data;
   Node *forw;
};

class DList
{
   private:
   Node *head;
   int count;
   Node *tail;

   public:
   bool swap(int n);
};
```