PRACTICE QUESTIONS

1. Write a function moveMaxToFront() that traverses a linked list of integers at most once, then moves the node with the largest stored value to the front of the list. The function prototype is given below:

```
int moveMaxToFront(ListNode  **ptrHead);
```

The function should return the original index of the node with the largest value if the operation was successful and -1 otherwise.

2. Write a function concatenate() that concatenates two linked lists. The function prototype is given below:

```
int concatenate(ListNode  **ptrHead1,
                ListNode  *head2);
```

The function should return 0 if it completes successfully and -1 otherwise. The final node of the first list should point to the first node of the second list. Your function should not create any new nodes. Be sure to handle the cases where one or more of the lists are empty.

3. Write a function combineAlternating() that combines two linked lists into a new one. Nodes from the two lists should be used in alternating order. The function prototype is given below:

```
int combineAlternating(ListNode  **ptrHead,
                       ListNode *head1,
                       ListNode  *head2);
```

The function should return 0 if it completes successfully and -1 otherwise.

Given the two lists [1 3 5 7] and [2 4 6 8], the new list created should be [1 2 3 4 5 6 7 8].

If one list is shorter than the other, nodes from the longer list should be used after the shorter list runs out. That is, given the two lists [1 3] and [2 4 6 8 10 12], the new list created should
be [1 2 3 4 6 8 10 12].

4. Write a function insertDbl() for a non-circular, doubly linked list of integers. The function prototype is given below:

```
int insertDbl(DblListNode  **ptrHead, int index, i nt value);
```

The function should return 0 if it completes successfully and -1 otherwise. Use the doubly linked list nodes as defined in lecture.
