CSC 520
Homework 2
Due: Wednesday September 15, 2010

Please make sure to express your algorithms in pseudo-code when directed, and always provide justification for your answer when asked to give the running time of an algorithm. Be brief and concise, and draw pictures where appropriate, (and, no, we don't expect you to actually submit any video presentations...)

1. Write an efficient algorithm to merge two sorted linked lists. What is the time complexity of your algorithm?

2. Swap two adjacent elements by adjusting only the pointers and not the data using
   a. Singly linked lists
   b. Doubly linked lists

3. Given a linked list L, and another linked list P containing integers in ascending order. The operation select_print(L, P) will print the elements in L that are in positions specified by P. For example, if P = 1, 3, 6, then the first, third and sixth elements in L are printed. Write the procedure select_print(L,P). What is the time complexity of your procedure?

4. An alternative to the deletion strategy to delete a node from a linked list is to use *lazy deletion*. To delete an element, we merely marked it deleted (using an extra bit field). The number of deleted and nondeleted elements in the list is kept as part of the data structure. If there are as many deleted elements as nondeleted elements, we traverse the entire list, perform the standard deletion algorithm on all marked nodes.
   a. List the advantages and disadvantages of lazy deletion.
   b. Write algorithms to implement the standard linked list operations (insert, find, delete) using lazy deletion.

5. Propose a data structure that supports the stack *push* and *Pop* operation and a third operation *find_min* which returns the smallest element in the data structure, all in O(1) time.

6. Implement a queue using two stacks efficiently. More specifically, you are given two stacks together with the PUSH and POP operations. However, it is not known to you how the stacks are implemented, that is, you may not assume that you have arrays or linked lists as their storage structures. Use these two stacks as the storage structure for a queue. Write efficient algorithms to INSERT and DELETE elements from the queue. The time complexity of your algorithm must be at most $O(m+n)$ for a total of n insertions and m deletions.

7. Find the computational complexity for the following:

a.
```
for ( ctn1 = 0, i = 1; i<= n; i++)
    for ( j = 1; j <= n; j++)
        cnt1++;
```

b.
```
for ( ctn2 = 0, i = 1; i<= n; i++)
    for ( j = 1; j <= i; j++)
        cnt2++;
```

c.
```
for ( ctn3 = 0, i = 1; i<= n; i = i * 2)
    for ( j = 1; j <= n; j++)
        cnt3++;
```

d.
```
for ( ctn4 = 0, i = 1; i<= n; i = i * 2)
    for ( j = 1; j <= i; j++)
        cnt4++;
```

e.
```
sum := 0;
for k := 1 to n do
    for j := 1 to k**2 do
        if (j mod k = 0) then
            for m := 1 to j do
                sum ;= sum + 1;
```