This problem set is individual and worth a total of 100 points. Solutions **must** be handed in as a hard copy at the beginning of class on 2/25. Please write your answers **clearly within the space provided** for each question.

Your Name: _____

1. Give the exact number of pointers present in Linked Lists of length n with the following characteristics (the total count may include `head`, `tail`, and links between nodes):
   (a) [5 points] A singly linked list with **no** `tail` pointer

   (b) [5 points] A singly linked list with a `tail` pointer

   (c) [5 points] A doubly linked list with **no** `tail` pointer

2. [5 points] Out of insertion sort and selection sort, which sorting algorithm has a better best-case runtime? What type of input would yield this best case behavior?

3. For a dynamic array with an initial capacity of 1, where elements are inserted one at a time:
   (a) [5 points] How many times would the array need to be resized to accommodate the insertion of 40 elements if its scaling factor was 3.0?

(b) [5 points] How many times would the array need to be resized to accommodate the insertion of 100 elements if its scaling factor was 2.0?

(c) [10 points] What is the general formula for how many times the array must be resized if it has a scaling factor of k and needs to accept n elements?

---

4. [30 points] Provide the time complexity, using Θ notation, of efficient algorithms for the functions below. Each column represents a linked list. All linked lists store a single integer at each node, no list stores its element count internally, and all lists have a tail pointer except when indicated.

Function	SLL	SLL (no tail)	DLL	CDLL
`int size()`				
`int at(int)`				
`int front()`				
`int back()`				
`bool empty()`				
`void clear()`				
`void set(int, int)`				
`void push_back(int)`				
`int pop_back()`				
`void insert_at(int, int)`				
`void delete_at(int)`				
`void reverse()`				

5. [10 points] Suppose that you have a SLL without a tail pointer. Write a $O(n)$ function to reverse the list. The class `SLL` contains *only* a single pointer `head`. Each `Node` in the list contains a pointer `next` to the next element.

```
void SLL::reverse() {



}
```

6. [5 points] What major advantages does a dynamic array have over a linked list?

7. [5 points] What major advantages does a linked list have over a dynamic array?

8. [10 points] Examine the following function which implements a destructor for a singly linked list class `SLL`. Is there a bug? If yes, how would you fix it?

```
SLL::~SLL() {
    Node *p = head;
    // go through list and remove all the nodes
    while(p != NULL) {
        delete p;
        p = next;
    }
}
```