Section #5

Tyler Conklin and Kate Rydberg (Based on handouts by previous CS106B instructors and TAs, especially those of Ashley Taylor and Shreya Shankar.)

Pointers, Arrays, and Classes

1. Pointer Mystery

The following code C++ uses pointers and produces two lines of output. What is the output?

```
void parameterMystery() {
    int v1 = 10;
    int v2 = 25;
    int* p1 = &v1;
    int* p2 = &v2;

    *p1 += *p2;
    p2 = p1;
    *p2 = *p1 + *p2;

    cout << v1 << " " << v2 << endl;
    cout << *p1 << " " << *p2 << endl;
}
```

2. Memory Trace and Structs

Given the following structs:

```
typedef struct Quidditch {
    int quaffle;
    int *snitch;
    int bludger[2];
} Quidditch;
```

```
typedef struct Hogwarts {
    int wizard;
    Quidditch harry;
    Quidditch *potter;
} Hogwarts;
```

Draw the memory diagram for the following pointer trace.

```
Quidditch * hufflepuff(Hogwarts * cedric) {
    Quidditch *seeker = &(cedric->harry);
    seeker->snitch = new int;
    *(seeker->snitch) = 2;
    cedric = new Hogwarts;
    cedric->harry.quaffle = 6;
    cedric->potter = seeker;
    cedric->potter->quaffle = 8;
    cedric->potter->snitch =
        &(cedric->potter->bludger[1]);
    seeker->bludger[0] = 4;
    return seeker;
}
```

```
void gryffindor() {
    Hogwarts *triwizard = new Hogwarts[3];
    triwizard[1].wizard = 3;
    triwizard[1].potter = NULL;
    triwizard[0] = triwizard[1];
    triwizard[2].potter = hufflepuff(triwizard);
    triwizard[2].potter->quaffle = 4;
}
```

3. Stretch (Arrays)

Write a member function named `stretch` that could be added to the `ArrayIntList` class. Your function accepts an integer `k` as a parameter and replaces each integer in the original list with k copies of that integer. For example, if an `ArrayIntList` variable named list stores this sequence of values:

{18, 7, 4, 4, 24, 11}

And the client makes the following call of `list.stretch(3);`, the list should be modified to store the following values:

{18, 18, 18, 7, 7, 7, 4, 4, 4, 4, 4, 4, 24, 24, 24, 11, 11, 11}

Notice that there are three copies of each value from the original list because 3 was passed as the parameter value.

If the value of k is less than or equal to 0, the list should be empty after the call.

Constraints: Do not call any member functions of the `ArrayIntList` (i.e., add, insert, remove, remove, size). You may, of course, refer to the private member variables inside the list. Do not make assumptions about how many elements are in the list or about the list's capacity. Do not use any auxiliary data structures to solve this problem (no vector, stack, queue, string, etc). You should not create any auxiliary arrays unless it is absolutely necessary to do so to solve the problem. Do not leak memory; if you cease using any dynamically allocated (via new) memory, free it. Your code must run in no worse than O(N) time, where N is the length of the list. Assume that you are adding this method to the `ArrayIntList` class as defined below:

```
class ArrayIntList {
private:
    int* elements;
    int mysize;
    int capacity;
public:
    ...
};
```

4. Mirror (Arrays)

Write a member function named `mirror` that could be added to the `ArrayIntList` class. Your function should double the size of the list of integers by appending the mirror image of the original sequence to the end of the list. The mirror image is the same sequence of values in reverse order. For example, suppose a variable named list stores the following values:

{1, 3, 2, 7}

If we make the call of `list.mirror();` then it should store the following values after the call:

{1, 3, 2, 7, 7, 2, 3, 1}

The list has been doubled in size by having the original sequence appearing in reverse order at the end of the list. If the list is empty, it should also be empty after the call. Assume that you are adding this method to the `ArrayIntList` class as defined in the previous problem.

5. Date (Classes)

Write a class named `Date` that remembers information about a month and day. Ignore leap years and don't store the year in your object. You should write both the .h and the .cpp files. You must include the following public members:

Member Name	Description
`Date(m, d)`	constructs a new date representing the given month and day
`daysInMonth()`	returns the number of days in the month stored by your date object
`getDay()`	returns the day
`getMonth()`	returns the month
`nextDay()`	advances the Date to the next day, wrapping to the next month and/or year if necessary
`toString()`	returns a string representation such as "07/04"