

Section #4 Solutions

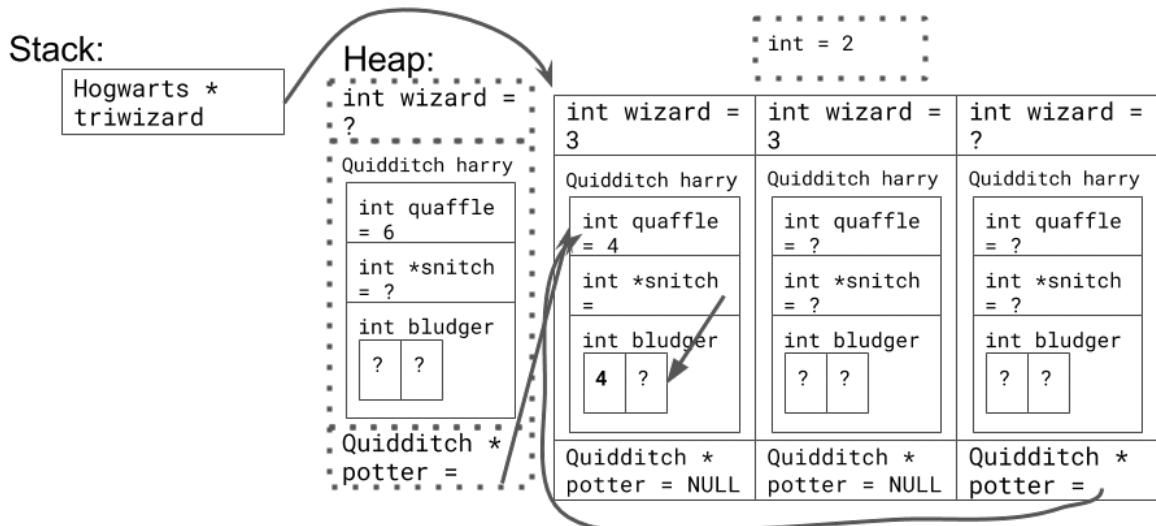
Based on handouts by various current and past CS106B/X instructors and TAs.

1. Pointer Trace (*pointers, dynamic memory, tracing*)

```
Line 1: 9 -3 1 0xcc00
Line 2: -7 9 6 0x555500
Line 3: 5 -7 2 0xdd00
Line 4: 5 0x555500 9 -3 -7 0xaa00 5
Line 5: 0xaa00 0xbb00 0xcc00 0xdd00 0xee00
```

2. Very Complex Pointer Trace (*pointers, dynamic memory, tracing*)

Orphaned memory is represented with dotted lines. For a full walkthrough of the solution, check out:
<http://tinyurl.com/106XHogwartsPointers>



3. Cleaning Up After Yourself (*dynamic memory, debugging*)

The first piece of code has two errors in it. First, the line

```
baratheon = targaryen;
```

causes a memory leak, because there is no longer a way to deallocate the array of three elements allocated in the first line. Second, since both `baratheon` and `targaryen` point to the same array, the last two lines will cause an error.

The second piece of code is perfectly fine. Even though we execute

```
delete[] stark;
```

twice, the array referred to each time is different. [Remember that you delete arrays, not pointers.](#)

Finally, the last piece of code has a double-delete in it, because the pointers referred to in the last two lines point to the same array.

4. Unit Tests (*testing*)

There are lots of good candidates for testing for the `Date` class! Below are a few examples.

```
ADD_TEST("Printing Date") {
    Date testDate(10, 22);
    expect (testDate.toString() == "10/22");
}

ADD_TEST("Advance a day in the same month") {
    Date testDate(10, 22);
    testDate.nextDay();
    expect (testDate.toString() == "10/23");
}

ADD_TEST("Advance a day into the next month") {
    Date testDate(10, 31);
    testDate.nextDay();
    expect (testDate.toString() == "11/1");
}

ADD_TEST("Advance a day into the next year") {
    Date testDate(12, 31);
    testDate.nextDay();
    expect (testDate.toString() == "1/1");
}

ADD_TEST("Days in month updates") {
    Date testDate(1, 31);
    int janDays = testDate.daysInMonth();
    testDate.nextDay();
    int febDays = testDate.daysInMonth();
    expect (janDays == 31 && febDays == 28);
}
```

Note that this list is not exhaustive. Other tests could involve testing `getDay()` or `getMonth()` as well.

5. First Date (*classes*)

<pre>// Date.h class Date { public: Date(int m, int d); int getDay() const; int getMonth() const; int getDaysInMonth() const; void nextDay(); string toString() const; private: int month; int day; // bonus int year; bool isLeapYear() const; };</pre>	<pre>// Date.cpp //all references to leap years are for the bonus static const HashSet<int> LONG_MONTHS{1,3,5,7,8,10,12}; Date::Date(int m, int d, /*bonus*/, int y) { month = m; day = d; // bonus year = y; } int Date::getDay() const { return day; } int Date::getMonth() const { return month; }</pre>
---	---

```

int Date::daysInMonth() const {
    if (month == 2) return 28;
    // bonus:
    // if (month == 2) return isLeapYear() ? 29 : 28;
    else if (LONG_MONTHS.contains(month)) return 31;
    else return 30;
}

void Date::nextDay() {
    day++;
    if (day > daysInMonth()) {
        month++;
        if (month > 12) {
            month++;
            // bonus
            year++;
        }
        day = 1;
    }
}

string Date::toString() const {
    return integerToString(month) + "/" +
           integerToString(day);
}

bool Date::isLeapYear() const {
    return year % 4 == 0 &&
           (year % 100 != 0 || year % 400 == 0);
}

```

6. IntArrayList (*classes, pointers, dynamic memory*)

```

// IntArrayList.h

class IntArrayList {
public:
    IntArrayList();
    ~IntArrayList();
    void add(int value);
    void insert(int index, int value);
    void clear();
    int get(int index) const;
    void set(int index, int value);
    int size() const;
    bool isEmpty() const;
    void remove(int index);
    string toString();
private:
    void expandCapacity();

    int* myElements; // array of elements
    int myCapacity; // length of array
    int mySize; // number of elements added
};

// IntArrayList.cpp implementation on next page

```

```

// IntArrayList.cpp
IntArrayList::IntArrayList() {
    myCapacity = 10;
    myElements = new int[myCapacity];
    mySize = 0;
}

IntArrayList::~IntArrayList() {
    delete[] myElements; //don't leak memory
}

// O(1) average case
void IntArrayList::add(int value) {
    if (mySize == myCapacity) {
        expandCapacity();
    }
    myElements[mySize] = value;
    mySize++;
}

// O(N) average case
void IntArrayList::insert(int index,
                           int value) {
    if (mySize == myCapacity) {
        expandCapacity();
    }
    for (int i = mySize; i > index; i--) {
        myElements[i] = myElements[i - 1];
    }
    myElements[index] = value;
    mySize++;
}

// O(1) average case
void IntArrayList::clear() {
    size = 0;
}

int IntArrayList::get(int index) const {
    return myElements[index];
}

void IntArrayList::set(int index,
                       int value) {
    myElements[index] = value;
}

int IntArrayList::size() const {
    return mySize;
}

```

```

bool IntArrayList::isEmpty() const {
    return mySize == 0;
}

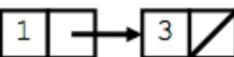
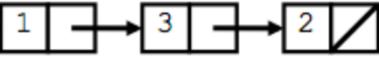
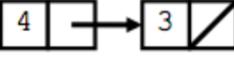
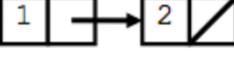
// O(N) average case
void IntArrayList::remove(int index) {
    for (int i = index; i < mySize - 1; i++) {
        myElements[i] = myElements[i + 1];
    }
    mySize--;
}

string IntArrayList::toString() {
    string result = "{";
    if (!isEmpty()) {
        result += integerToString(myElements[0]);
        for (int i = 1; i < mySize; i++) {
            result += "," + integerToString(myElements[i]);
        }
    }
    result += "}";
    return result;
}

void IntArrayList::expandCapacity() {
    myCapacity *= 2;
    int* biggerArray = new int[myCapacity];
    for (int i = 0; i < mySize; i++) {
        biggerArray[i] = myElements[i];
    }
    delete[] myElements; //don't leak memory
    myElements = biggerArray;
}

```

7. Linked Nodes (*pointers*)

- a) 
- b) 
- c) 
- d) 

8. Linked Nodes 2 (*pointers*)

a)

```
ListNode node = {3, nullptr};  
list->next->next = &node;
```

b)

```
ListNode node = {3, list};  
list = &node;
```

c)

```
temp->next->next = list->next;  
list->next = temp;
```

d)

```
list->next->next = temp->next;  
temp->next = list->next;  
list->next = temp;
```

e)

```
ListNode* list2 = list;  
list = list->next;  
list2->next = list2->next->next;  
list->next = nullptr;
```

f)

```
ListNode* temp = list->next->next;  
temp->next = list->next;  
list->next->next = list;  
list->next->next->next = nullptr;  
list = temp;
```

g)

```
list->next->next->next = list;  
list = list->next->next;  
ListNode* list2 = list->next->next;  
list->next->next = nullptr;
```