CS 106B Autumn 2015 Midterm Exam
ANSWER KEY

1. Parameters and Pointers (read)

```
203 3005 11
3003 16 204
11 204 3003 16
```

2. Collections (read)

```
a)
before: {1, 2, 3, 4, 5}
after : {1, 2, 3, 5, 2, 4, 1}

b)
before: {67, 29, 115, 84, 33, 71, 90}
after : {29, 33, 90, 5, 71, 4, 84, 3, 115, 2, 67, 1}
```

3. Collections (write)

Every programming problem can be solved in multiple ways.

```
// solution 1
void areaCodes(string filename) {
    // open file
    ifstream input;
    input.open(filename.c_str());
    if (input.fail()) { return; }

    // read file data into map of sets
    Map<string, Set<string> > numbers;
    string line;
    while (getline(input, line)) {
        string areaCode = line.substr(0, 3);
        numbers[areaCode].add(line);
    }

    // find most popular area code
    string best = "";
    for (string areaCode : numbers) {
        if (best.empty() || numbers[areaCode].size() > numbers[best].size()) {
            best = areaCode;
        }
    }

    // print all numbers in that area code
    for (string number : numbers[best]) {
        cout << number << endl;
    }
}
```

4. Big-Oh (read)

i. e) $O(N^2)$
ii. d) $O(N \log N)$
iii. f) $O(N^2 \log N)$
iv. e) $O(N^2)$

5. Recursion (read)

```
a) recursionMystery9(12, 49)      = 1429
b) recursionMystery9(73, -8)      = -7038
c) recursionMystery9(-248, -3795) = 3274985
```

6. Recursion (write)

```
// solution 1
string replaceAll(string s, char from, char to) {
    if (s.empty()) {
        return s;
    } else {
        char first = s[0];
        string rest = s.substr(1);
        if (first == from) {
            first = to;
        }
        return first + replaceAll(rest, from, to);
    }
}

================================================================================

// solution 2
string replaceAll(string s, char from, char to) {
    if (s.empty()) {
        return s;
    } else if (s[0] == from) {
        return to + replaceAll(s.substr(1), from, to);
    } else {
        return s[0] + replaceAll(s.substr(1), from, to);
    }
}
```

7. **Backtracking (write)**

```
// solution 1
void phoneHelper(string phoneNumber, Lexicon& dictionary,
                 Map<int, string> letterMap, string chosen) {
    if (dictionary.containsPrefix(chosen)) {
        if (phoneNumber == "") {
            cout << chosen << endl;
        } else {
            int digit = phoneNumber[0] - '0';
            string rest = phoneNumber.substr(1);
            string letters = letterMap[digit];
            for (int i = 0; i < letters.length(); i++) {
                phoneHelper(rest, dictionary, letterMap, chosen + letters[i]);
            }
        }
    }
}

void phoneWords(string phoneNumber, Lexicon& dictionary, Map<int, string>& letterMap) {
    phoneHelper(phoneNumber, dictionary, letterMap, "");
}
```

8. **Pointers and Linked Nodes (write)**

```
// solution 1
list2->next->next = list2;          // 3 -> 2               // reverse 3 and 2
list2 = list2->next;                // list2 -> 3
list2->next->next = NULL;           // 2 /
list1->next = new ListNode(4);      // 1 -> 4               // insert 4
ListNode* temp = list1;             // temp -> 1 /          // swap list1 and list2
list1 = list2;                      // list1 -> 3
list2 = temp;                       // list2 -> 4


// solution 2
ListNode* temp = list1;          // temp -> 1      // swap list1 and list2
list1 = list2;                   // list1 -> 2
list2 = temp;                    // list2 -> 1
list1->next->next = list1;       // 3 -> 2         // reverse 3 and 2
list1 = list1->next;             // list1 -> 3
list1->next->next = NULL;        // 2 /
list2->next = new ListNode(4);   // 1 -> 4
```