CS 106B Practice Midterm Exam #1
ANSWER KEY

1. C++ Basics / Parameters

```
9 -3 1 0xcc00
-7 9 6 0xbb00
5 -7 2 0xdd00
5 9 -3 -7
```

2. File I/O and Strings (write)

As with any programming problem, there are many correct solutions. Here is one:

```
void wordStats(string filename) {
    ifstream input;
    input.open(filename);
    if (input.fail()) {
        cout << "Error, bad input file." << endl;
        return;
    }

    HashSet<char> uniqueLetters;
    int charCount = 0;
    int wordCount = 0;
    string word;
    while (input >> word) {
        wordCount++;
        charCount += word.length();
        for (int i = 0; i < word.length(); i++) {
            char ch = tolower(word[i]);
            if (isalpha(ch)) {
                uniqueLetters.add(ch);
            }
        }
    }

    cout << "Total words    = " << wordCount << endl;
    cout << "Average length = " << ((double) charCount / wordCount) << endl;
    cout << "Unique letters = " << uniqueLetters.size() << endl;
}
```

3. ADTs / Collections (read)

Vector	Output
a) {5, 2, 5, 2}	{2, 2}
b) {3, 5, 8, 9, 2}	{5, 9, 1, 3}
c) {0, 1, 4, 3, 1, 3}	{1, 3, 3, 1}

4. ADTs / Collections (write)

Here are two working solutions:

```
Map<int, string> byAge(const Map<string, int>& ages, int min, int max) {
    Map<int, string> result;
    for (string name : ages) {
        int age = ages[name];
        if (min <= age && age <= max) {
            if (result.containsKey(age)) {
                string value = result.get(age);
                value += " and " + name;
                result.put(age, value);
            } else {
                result.put(age, name);
            }
        }
    }
    return result;
}
```

```
Map<int, string> byAge(const Map<string, int>& ages, int min, int max) {
    Map<int, string> result;
    for (string name : ages) {
        if (min <= ages[name] && ages[name] <= max) {
            if (result.containsKey(ages[name])) {
                result[ages[name]] += " and ";
            }
            result[ages[name]] += name;
        }
    }
    return result;
}
```

5. Big-Oh Analysis (read)

a) $O(N)$
b) $O(N^2)$
c) $O(1)$
d) $O(N^3)$
e) $O(N \log N)$

6. Recursion (read)

Call	Result
a) recursionMystery1(6, 3);	6 0 3
b) recursionMystery1(2, 3);	2 0 1 3
c) recursionMystery1(5, 8);	5 2 0 1 3 8
d) recursionMystery1(21, 12);	21 9 6 0 3 12
e) recursionMystery1(3, 10);	3 2 0 1 4 7 10

7. Recursion (write)

Here are two working solutions:

```
string moveToEnd(string s, char c) {
    if (s.length() == 0) {
        return "";
    } else if (toupper(s[0]) == toupper(c)) {
        char upperC = toupper(s[0]);
        return moveToEnd(s.substr(1), c) + upperC;
    } else {
        return s[0] + moveToEnd(s.substr(1), c);
    }
}

string moveToEnd(string s, char c) {
    if (s == "") {
        return s;
    }
    char first = s.at(0);
    s = s.substr(1, s.length() - 1);
    s = moveToEnd(s, c);
    if (toupper(first) == toupper(c)) {
        char upperFirst = toupper(first);
        return s + upperFirst;
    } else {
        return first + s;
    }
}
```

8. Recursive Backtracking (write)

```
string crack(int maxLength) {
    if (maxLength < 0) {
        throw maxLength;
    }
    string result;
    for (int length = 1; length <= maxLength; length++) {
        if (crackHelper(result, "", length)) {
            return result;
        }
    }
    return "";
}

bool crackHelper(string& result, string chosen, int length) {
    if (length == 0) {
        if (isCorrectPassword(chosen)) {
            result = chosen;
            return true;
        } else {
            return false;
        }
    } else {
        for (char c = 'a'; c <= 'z'; c++) {
            if (crackHelper(result, chosen + c, length - 1)) {
                return true;
            }
        }
        return false;
    }
}
```

9. Implementing a Collection Class (write)

```
int ArrayList::maxCount() const {
    if (mysize == 0) {
        return 0;
    } else {
        int max = 1;
        int count = 1;
        for (int i = 1; i < mysize; i++) {
            if (elements[i] == elements[i - 1]) {
                count++;
                if (count > max) {
                    max = count;
                }
            } else {
                count = 1;
            }
        }
        return max;
    }
}
```

10. Pointers and Linked Nodes (write)

```
list2->next->next->next = list;     // 4 -> 1
list->next = list2;                 // 1 -> 2
list = list2->next->next;           // list -> 4
list2 = list2->next;                // list2 -> 3
list2->next = NULL;                 // 3 /
list->next->next->next = NULL;      // 2 /
```