Test 2: CPS 6

50 Minute Exam

March 31, 1999

Name (print): ______________________________ Lab # _____

Honor Acknowledgment (signature): ______________________________

DO NOT SPEND MORE THAN 10 MINUTES ON ANY OF THE QUESTIONS! If you do not see the solution to a problem right away, move on to another problem and come back to it later.

Before starting, make sure your test contains 10 pages.

If you think there is a syntax error, then ask. You may assume any include statements are provided.

<table>
<thead>
<tr>
<th>Problem</th>
<th>value</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>14 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 2</td>
<td>16 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 3</td>
<td>10 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 4</td>
<td>20 pts.</td>
<td></td>
</tr>
<tr>
<td>Optional</td>
<td>(6) pts.</td>
<td></td>
</tr>
<tr>
<td>TOTAL:</td>
<td>60 pts.</td>
<td></td>
</tr>
</tbody>
</table>
PROBLEM 1: \((\text{Trouble?}: (14 \text{ pts}))\)

Part A: (6 points)
Consider the function \(\text{Trouble}\) below.

\[
\text{int Trouble (const Vector<int> & numbers, int size)}
\{
    \text{if (size == 1)}
    \{
        \text{return numbers[0];}
    \}
    \text{else}
    \{
        \text{return Trouble(numbers, size - 1) - numbers[size - 1];}
    \}
\}
\]

Draw the recursive calls (and their return values) generated by calling \(\text{Trouble(numbers, 4)}\), where \(\text{numbers}\) contains the following values:

5 2 3 1
Part B: (8 points)

Consider the function *DoubleTrouble* below.

```c
int DoubleTrouble (const Vector<int> & numbers, int start, int end)
{
    if (start >= end)
    {
        return numbers[start];
    }
    else
    {
        int mid = (start + end) / 2;
        return DoubleTrouble(numbers, start, mid) + DoubleTrouble(numbers, mid + 1, end);
    }
}
```

Draw the recursive calls (and their return values) generated by calling *DoubleTrouble(numbers, 0, 2)*, where *numbers* contains the following values:

```
5 2 3
```
PROBLEM 2:  (Reversi: 16 points)

For this problem, you will be printing out the words in a file in reverse order. The first part asks you to do it iteratively; the second part, recursively.

For example, if a file contained the words:

I can read.

your functions for both parts should generate the output below:

read.
  can
  I

Part A: (8 points)

Complete the function PrintWordsReversed which is started below. PrintWordsReversed prints out the words in the given file in reverse order (one per line). For this part of the problem, you must implement your function iteratively (i.e., with a loop).

In order to do this, you must first save them in a vector as you read them (this has already been done). Then, to print the words backwards, start at the back of the vector and move forwards, printing each word as you go.

Complete the function PrintWordsReversed below.

    void PrintWordsReversed (ifstream & input)
    // pre:  input is open and ready for reading
    // post: words in input are printed in reverse order
    {
      Vector<string> words;

      string word;
      while (input >> word)
      {
        words.append(word);
      }
    }
Part B: (8 points)

Complete the function `PrintWordsReversedR` which is started below. `PrintWordsReversedR` prints out the words in the given file in reverse order (one per line). For this part of the problem, a recursive implementation that reads the words from the file has been written already.

You must complete the function by adding code in the appropriate place to print each word during the recursion so they appear in reversed order when the first call to `PrintWordsReversedR` is finished. Complete the function `PrintWordsReversedR` below.

```cpp
void PrintWordsReversedR (ifstream & input)
// pre: input is open and ready for reading
// post: words in input are printed in reverse order
{
    string word;

    if (input >> word)
    {

        PrintWordsReversedR(input);

    }
}
```
PROBLEM 3: (What’s bugging you: 10 points)

Consider the following function (lines are numbered for easier reference):

```
1: void DoubleItUp (Vector<string> & words, int size)
2:     // pre: words contains size elements, for example, size = 3, words = (a,b,c)
3:     // post: words contains 2*size elements, such that words = (a,a,b,b,c,c)
4: {
5:     // make room for new elements
6:         words.SetSize(2 * size);
7:     
8:     // copy each element twice
9:         int k;
10:    for (k = 0; k < size; k++)
11:        {
12:             words[2 * k + 1] = words[k];
13:             words[2 * k] = words[k];
14:        }
15: }
```

The intent of the function is to "double" a vector by replacing every element in the vector with two identical elements. For example, doubling the following vector of strings

```
"apple"  "orange"  "pear"
```

should change the vector as shown below:

```
"apple"  "apple"  "orange"  "orange"  "pear"  "pear"
```

However, as written, the code changes the vector in following manner:

```
"apple"  "apple"  "apple"  "apple"  "apple"  "apple"
```

Explain what the error in the program is (i.e., why it behaves the way it does) AND how the error can be corrected.
PROBLEM 4:  (Remember albums?: (20 pts))

For this problem, information about an album is stored in a struct as defined below:

```cpp
struct AlbumInfo
{
    string artist;
    string title;

    AlbumInfo (const string & anArtist, const string & aTitle);
};

AlbumInfo::AlbumInfo (const string & anArtist, const string & aTitle)
{
    artist = anArtist;
    title = aTitle;
}
```

Part A: (5 points)
Write the function `PrintAlbumsByArtist` whose header is given below. `PrintAlbumsByArtist` should print the titles of only those albums by the given artist. Each title should be printed on a line by itself, indented by two spaces.

Complete the function `PrintAlbumsByArtist` below.

```cpp
void PrintAlbumsByArtist (Vector<AlbumInfo> & albums, const string & artist)
// pre: albums contains albums.size() elements
// post: all album titles by artist are printed
{
    cout << artist << endl;
```
Part B: (5 points)

Complete the function `LoadAlbums` which is started below. `LoadAlbums` reads a given file line by line and constructs an album from the data in each line. You may assume there are no errors in the format of the data file.

For this part, you do not need to know the format of the data file. All you need to know is that all the information for an album is given on a single line. Use the function `LoadAlbum` to read a line of data and return an `AlbumInfo` struct. For this problem, you do NOT need to implement the function `LoadAlbum`.

```cpp
AlbumInfo LoadAlbum (istringstream & inputLine)
// pre: inputLine contains album data in the proper format
// post: returns album with the artist and title data in inputLine
```

Complete the function `LoadAlbums` below.

```cpp
void LoadAlbums (ifstream & input, Vector<AlbumInfo> & albums)
// pre: input is open and ready for reading
// post: albums is filled with the artist and title data from each
//       line in input, it contains albums.size() elements
{
    string line;

    while (getline(input, line))
    {
```
**Part C: (10 points)**

Complete the function `LoadAlbum` which is started below. `LoadAlbum` should read the artist and title of a single album from the given stream. You may assume that the given data is correctly formatted, the artist information comes first and is separated from the title by a colon. Both the artist and title can be one or more words long, but are guaranteed not to contain a ";".

For example a stream might contain the following data about an album:

Bee Gees : Saturday Night Fever

The function should return an `AlbumInfo` struct containing the following information:

<table>
<thead>
<tr>
<th>Artist</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Bee Gees&quot;</td>
<td>&quot;Saturday Night Fever&quot;</td>
</tr>
</tbody>
</table>

Complete the function `LoadAlbum` below.

```cpp
AlbumInfo LoadAlbum (istream & inputLine)
// pre:  inputLine contains album data in the proper format
// post: returns album with the artist and title data in inputLine
{
    string artist, title;

    string word;
    while (inputLine >> word)
    {
        // process each word here
    }

    // return the AlbumInfo struct
    return AlbumInfo(artist, title);
}
```
PROBLEM 5: (A man, a plan, a canal — Panama! (6 points))

EXTRA CREDIT **** OPTIONAL ***** EXTRA CREDIT

Write the function `IsPalindrome` whose header is given below. `IsPalindrome` should return true only if the given string is a palindrome. A palindrome is a word or phrase that can be read identically forward or backward (ignoring punctuation, capitalization, and spacing).

For example, calling `IsPalindrome("racecar")` should return true. Likewise, calling `IsPalindrome("amanaplanacanalpanama")` should return true. Calling `IsPalindrome("motor")` should return false.

Complete the function `IsPalindrome` below.

```cpp
bool IsPalindrome (const string & phrase) {
    // pre: phrase does not contain whitespace, capitals, or punctuation
    // post: returns true if phrase is a palindrome,
    //       false otherwise
    
```