NAME (print): ___________________________________ Lecture # _____ Lab # ______

Honor Acknowledgment (signature): _______________________________________

DO NOT SPEND MORE THAN 10 MINUTES ON ANY OF THE OTHER QUESTIONS! If you
don’t 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>12 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 2</td>
<td>14 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 3</td>
<td>12 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 4</td>
<td>20 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 5</td>
<td>16 pts.</td>
<td></td>
</tr>
<tr>
<td>Optional</td>
<td>(8) pts.</td>
<td></td>
</tr>
<tr>
<td>TOTAL:</td>
<td>74 pts.</td>
<td></td>
</tr>
</tbody>
</table>

Selected member functions of the string class

class string{
    public:
        int length(); // returns length of string
        string substr(int positn, int leng); // returns portion of string
            // starting at positn with length leng
        int find(string s); // returns starting position of s in string
            // and returns NPOS if s not found in string
};
**PROBLEM 1:**  (What DID you do?: (12 pts))

**Part A:** (6 points)

Indicate the output of the cout statements below. Assume the statements are part of a program that compiles and runs. You can show your reasoning for partial credit. (Warning: the output isn’t supposed to make a lot of sense.)

Show blanks by putting an “_” for each blank printed.

```c++
string other = "Persnickity";
int num = 9;

if (5 > num)
    cout << "quack " << endl;
else if (15 > num)
    cout << "quack " + "quack " << endl;
else if (25 > num)
    cout << "quack " + "quack " << "quack " << endl;
else
    cout << "quack " + "quack " << "quack " + "quack " << endl;
cout << "number " << num << endl;
if (15 > num) // watch out
    cout << other.substr(4,4) << endl;
    cout << other << endl;
if (35 > num)
    cout << other.substr(7,3) << endl;
else
    cout << other.substr(0,5) << endl;
```
Part B: (6 points)
Indicate the output for the following program. Note the use of reference parameters.

```c
int funcA(int x)
{
    x += 7;
    return x;
}

void funcB(int & y)
{
    y = 42;
}

void funcC(int z, int & w)
{
    z = 0;
    w = 0;
}

int main()
{
    int r=13, s=11;
    cout << r+s << " r + s" << endl;
    r = funcA(s);
    cout << r << " " << s << endl;
    funcB(s);
    cout << r << " " << s << endl;
    funcC(r, s);
    cout << r << " " << s << endl;

    return 0;
}
```
PROBLEM 2: (Renew your magazines: 14 points)

Part A: (6 points)
Write the function FormLetter whose header is given below. FormLetter prints out a letter informing a given person that the given magazine subscription will expire on the first of the given year. For example, the following calls:

    FormLetter("Kenny", "Science", 1999);
    FormLetter("Lisa", "Games", 2002);

should generate the output below:

    Dear Kenny,
    Please note that your subscription to Science will expire on January 1, 1999. Kenny, please renew now!
    
    Dear Lisa,
    Please note that your subscription to Games will expire on January 1, 2002. Lisa, please renew now!

Complete the function FormLetter below.

    void FormLetter(string name, string magazine, int year)
    // post: form letter is printed out to terminal output: cout
    {
    }
Part B: (8 points)
Write the function `PrintFormLetters` whose header is given below. `PrintFormLetters` is given the name of a file containing name, magazine, and ending year information for each subscriber. In particular, each line of the file contains the name of the subscriber, followed by the name of the magazine subscribed to, and finally, the year the subscription for that magazine ends.

For example, suppose a data file contains the following information; then your function should generate the same output as the two calls shown in Part A.

```
Kenny Science 1999
Lisa Games 2002
```

You may assume that each entry is correctly formatted, all names are only one word, and always in the correct order.
You MUST use the function `FormLetter` from the previous part. Assume that it works correctly even if your implementation does not.

Complete the function `PrintFormLetters` below.

```cpp
void PrintFormLetters(string fileName)
{// pre: fileName is the name of a valid file
 // post: prints a form letter for each subscriber listed in fileName
{
    ifstream input;
    input.open(fileName);
```
PROBLEM 3:  (Squash those bugs: 12 points)

Consider the following program with lines numbered (#includes are not shown). The intent of the program is to print out the odd numbers between 1 and 100. However, there is a bug in the program as written and it generates the (abbreviated) output shown below:

1 is a number under 100
3 is a number under 100
5 is a number under 100
...
97 is a number under 100
99 is a number under 100
101 is a number under 100
103 is a number under 100
... (forever)

5: void PrintOddNumbers(int limit)
6: {
7:     int oddnum = 1;
8:     while (oddnum != limit)
9:         {
10:             cout << oddnum 
11:                 << " is a number under " << limit
12:                 << endl;
13:             oddnum += 2;
14:         }
15: }
16:
17: int main()
18: {
19:     PrintOddNumbers(100);
20:     return 0;
21: }

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:**  *(Running uphill (20 points))*

Write the function `NumClimbing` whose header is given below. `NumClimbing` returns the length of the longest run of non-decreasing digits within a given number. For example:

```cpp
cout << NumClimbing(31232323) << endl;
cout << NumClimbing(321) << endl;
cout << NumClimbing(66789) << endl;
cout << NumClimbing(3159265) << endl;
```

should generate the output below:

```
3
1
5
3
```

Remember that `num % 10` gives you the last digit, while `num / 10` gives you all the digits but the last. Complete the function `NumClimbing` below.

```cpp
int NumClimbing(int num)
// pre:  0 < num
// post: returns length of the longest run of number in non-decreasing order
{

    while (num > 0)
    {

        num /= 10;
    }
```
PROBLEM 5:  (*it’s full of stars!* (16 pts))

A class *Star* for manipulating information about our brightest stars is described below. For this problem, assume that the class includes information about all of the named stars and that the user will never ask about stars not included (nor misspell their names).

In astronomy, stars are noted (among other things) for their *magnitude* which is their brightness as seen from earth and their *color*. Note: the LOWER THE MAGNITUDE, THE BRIGHTER THE STAR !! Visible stars range in magnitude from -1.46 (brightest) to 6 (darkest visible without aids).

Here is the definition of the *Star* class.

```cpp
// class for manipulating stars

// Star(string starName) -- constructs the star named in the string
//   e.g., Star star("Sirius") creates
//   star (magnitude = -1.46) (color = "white")
// Star stern("Vega") creates
//   stern (magnitude = 0.03) (color = "blue-white")
//
// double Magnitude() -- returns magnitude of the current star
//   e.g 2.0 for the star Polaris
//
// string Color() -- returns color of the current star
//   e.g. "orange" for the star Betelgeuse
//
// string Constel() -- returns constellation of the current star
//   e.g. "Gemini" for the star Castor
//

class Star
{
    public:
        Star(string); // constructor, initial focus on star named
double Magnitude(); // returns magnitude of star
        string Color(); // returns color of star
        string Constel(); // returns constellation star is in

    private:
        // rest not shown (and not useful)
};
```
Part A: (4 points)
Write the function Brighter whose header is given below. The function should return the star that is brighter, i.e., has the smaller magnitude.

Star Brighter(Star starA, Star starB)
// post: returns Star with smaller magnitude
{

Part B: (12 points)
Write the function Brightest whose header is given below. Brightest is given a stream of star names separated by whitespace (as if from a file or from the terminal input) and should return the brightest star from those input. You may assume that each star name is only one word and can be used to construct a Star.

You must use Brighter from the previous part. You may assume that it works correctly even if your implementation does not.

Complete the function Brightest below.

Star Brightest(ifstream & input)
// pre: input is open and ready for reading
// post: returns the brightest Star from input
{

PROBLEM 6: (Who needs prompting: (8 points))
EXTRA CREDIT **** OPTIONAL ***** EXTRA CREDIT

Write the function PromptRange whose header is given below. PromptRange should prompt the user to enter a number within a given range. The prompt should be repeated until the user enters a number within the range. For example, the following calls:

```cpp
int days = PromptRange("Enter number of days", 1, 31);
int people = PromptRange("Please input the number of people", 0, 42);
```

should generate the output below (given the user enters these numbers):

```
Enter number of days between 1 and 31: 7
Please input the number of people between 0 and 42: 43
Please input the number of people between 0 and 42: -1
Please input the number of people between 0 and 42: 0
```

Complete the function PromptRange below.

```cpp
int PromptRange(string prompt, int low, int high)
// pre: low <= high
// post: returns a value between low and high (inclusive)
{
```