PROBLEM 1:  (Output: (12 pts))

Indicate the output of each of the `cout` statements below. Assume each statement is part of a program that compiles and runs. You can show your reasoning for partial credit.

**Show blanks** by putting an "_" for each blank printed.

```cpp
string word;
word = "favorite class";
int num = 9;

cout << "Hello" << word + word << endl;
cout << num/2 << "+" << num % 4 << endl;
cout << word.substr(word.find("ri"),4) << endl;
cout << 9 << "9" << endl << 6 + 2 * num << endl;
cout << word.length() << " " << num < 6 << endl;
```

**Output**

PROBLEM 2:  (Join Club: (12 pts))

The National Sports Association (NSA) allows sport clubs to join if they have at least 30 members. Since they get a lot of mail from clubs that want to join, they have created the following form letter that either tells a club they can join, or tells a club why they cannot join.

You should write a function named `JoinClub` that accomplishes this task. The two function calls below should generate the output shown.

```cpp
JoinClub("Duke","John Smith", 100);
JoinClub("NC State", "Mary", 18);
```

**Corresponding Output**

Dear John Smith,

Your sport club from Duke with 100 members
is welcome to join the NSA.

Dear Mary,
Your sport club from NC State with 18 members has too few members to join the NSA.

Write the function JoinClub below.

**PROBLEM 3:** *(Which Professor: (12 pts))*

Write the function *NumberToProf* whose header is given below. Given a CPS course number, *NumberToProf* returns the name of the Professor teaching the course in the spring semester. Valid course numbers and Professors are shown below. If the course number is not shown below, then *NumberToProf* returns "unknown".

<table>
<thead>
<tr>
<th>COURSE</th>
<th>CPS 1</th>
<th>CPS 6</th>
<th>CPS 100</th>
<th>CPS 104</th>
<th>CPS 108</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROFESSOR</td>
<td>Ramm</td>
<td>Ramm</td>
<td>Astrachan</td>
<td>Kedem</td>
<td>Astrachan</td>
</tr>
</tbody>
</table>

For example, *NumberToProf*(108) returns "Astrachan" and *NumberToProf*(103) returns "unknown".

Complete *NumberToProf* below the following header.

```cpp
string NumberToProf(int number)  
// postcondition: returns the name of the Professor teaching the CPS course 
// in the spring semester. If the course number is not valid, 
// returns "unknown"
{
}
```

**PROBLEM 4:** *(The Largest Roll: (12 pts))*

Write the function *LargestRoll* whose header is given below. *LargestRoll* continues to roll a pair of 6-sided dice until doubles is rolled, and then returns the value of the largest roll (sum of two dice in one roll). The Dice class is shown below.

```cpp
class Dice{
    public:
        Dice(int sides); // constructor
        int Roll(); // return the random roll
        int NumSides(); // how many sides this die has
        int NumRolls(); // # times this die rolled
    private:
        // not shown
};
```

Here is an example where four rolls were necessary. The first roll gives 4 and 1 for a sum of 5, the second roll gives 3 and 5 for a sum of 8, the third roll gives 3 and 2 for a sum of 5, and the fourth
roll finally is a double roll, 2 and 2, with a sum of 4. LargestRoll would return 8 in this case (the value of the rolls were, 5, 8, 5, and 4 and 8 is the value of the largest roll).

Complete LargestRoll below the following header.

```cpp
int LargestRoll()
// postcondition: returns the value of the largest roll once doubles is
// rolled.
{
}
```

PROBLEM 5:  (Where do they differ?: (12 pts))

Write the function PositionDiffer whose header is given below. PositionDiffer takes two integers of the same length and returns the first position number from the right where the two numbers differ. You can assume that both numbers are greater than 0 and do not contain the digit 0 in them.

For example, PositionDiffer(8746, 5546) returns 3 since the digits in the first positions (digit 6) and second positions (digit 4) are the same in both numbers and the digits in the third position (digits 7 and 5) differ. PositionDiffer(7546231, 8946231) returns 6 since the first two digits from the right that differ are 5 and 9 in the 6th position, and PositionDiffer(56,56) returns 0 since none of the digits in the same position differ.

Complete PositionDiffer below the following header.

```cpp
int PositionDiffer(int num1, int num2)
// precondition: num1 > 0, num2 > 0, num1 and num2 have the same number
// of digits and do not contain the digit 0.
// postcondition: returns the first position number from the right in which
// the two numbers differ, returns 0 if the two numbers are
// identical
{
}
```

PROBLEM 6:  (I can’t Drive, 55: (16 pts))

A class Car is designed for creating a car and driving it between cities. One can set or obtain information about the odometer, the device for keeping track of how many miles the car has been driven.

Below is an example usage of this class.

```
Car Honda("Raleigh");
cout << "Current odometer setting is " << Honda.ReadOdometer() << endl;
Honda.TravelToCity("NewYork");
cout << "Distance to New York is " << Honda.ReadOdometer() << " miles." << endl;
```
The output corresponding to this example is below. Note that the mileage between cities is stored inside the class and you don't need to know how it is stored in order to access this information.

Current odometer setting is 0
Distance to New York is 650 miles.

Here is the definition of the Car class.

```cpp
// Car class
//
// Car(string city) -- constructs a car in a particular city with
// initial odometer reading of 0 miles
//
// void SetOdometer(int num) -- Sets the odometer to a specific number
//
// int ReadOdometer() -- returns the current value of the Odometer
//
// void TravelToCity(string city) -- drives the car to the named city

class Car
{
    public:
        Car(string); // constructor, sets odometer to 0
        void SetOdometer(int); // set odometer to int value
        int ReadOdometer(); // returns odometer reading
        void TravelToCity(string); // drives car to named city

    private:
        void AddOdometer(int); // add int value to odometer
        int odometer; // current value of odometer
        // rest not shown
};
```

**Part A** 2 pts

Explain what is wrong with the following statements.

```cpp
Car Ford("Chicago");
Ford.SetOdometer(300);
cout << "Current odometer reading is " << Ford.odometer << endl;
Ford.AddToOdometer(1000);
cout << "Current odometer reading is " << Ford.odometer << endl;
```

**Part B** 14 pts

Write the function *PrintDistance* whose header is given below. *PrintDistance* does the following:

- Asks the user to type in the number of cities, must be &ge; 2
Asks the user to type in cities. Each city must be one word (contain no blanks). You can assume the Car class knows about all cities.

Creates a car that then drives the route of the list of cities printing the distance between each adjacent pair of cities.

For example, if the user typed the following input:

4
Raleigh
Durham
Cary
Wilmington

Then the output of PrintDistance (mixed with the user’s input) would be:

Enter number of cities: 4
Enter cities:
Raleigh
Durham
Distance between Raleigh and Durham is 28 miles.
Cary
Distance between Durham and Cary is 24 miles.
Wilmington
Distance between Cary and Wilmington is 112 miles.

Complete PrintDistance on the next page.

```c
void PrintDistance()
{

}
```