Below is descriptions for the 8 recursive functions. Included in this folder is the Driver.cpp file you should use to test your functions. Note that you may also put implementation of 8 functions at the end of the file rather than creating a separate .cpp file in which to put implementation of functions. Two of the functions must be implemented in Driver.cpp directly (or indirectly through an #include statement) because they are template functions.

**Write separate recursive functions to do the following:**

1) Convert the following function to one that uses recursion.

void sign(int n)

{

while (n >0) {

cout << “ No parking\n”;

n--;

}

2) Write a recursive function named addArray that accepts 3 arguments: an array of values and an int indicating the number of elements. The function should recursively calculate the sum of all the values in the array. Note1 : To make your solution more generic, make the function a template function that will accept an array<> of any type and any size. Note2: Implementation should be put in Driver.cpp because it is a template function.

3) Write a recursive function named isMember that accepts 3 arguments: an array of values, an int representing the size of the array and a single value. The function should return true if the single value is found in the array of values and false otherwise. Note1 : To make your solution more generic, make the function a template function that will accept an array<> of any type and any size. Note2: Implementation should be put in Driver.cpp because it is a template function.

4) How many possible bridge hands are there? This question is a specific case of the general question: how many combinations of X items can I make out of Y items? In the case of the bridge hand, x is 13 and y is 52. The solution is given by the following formula:

Combinations(Y, X) = Y ( if X == 1 )

Combinations(Y, X) = 1 ( if X == Y )

Combinations(Y, X) = Combinations(Y-1, X-1) + Combinations(Y-1, X) (if Y > X > 1 )

Write a recursive function that calculates the number of combinations of X items that can be made from Y items. Use it to answer the original question. Note: Your function will need to return type long long in order to represent the answer without overflow. My solution returns unsigned long long which is better again (because it can calculate twice as large a value). Note also that my solution take more than 20 minutes to compute. A non-recursive solution would take significantly less time to compute!

5) Write a recursive function to convert a number in base 10 to a number in base 2. The basic algorithm to do this is to keep dividing the number by 2 and output the remainder. The remainders need to be output in reverse order as computed. So, for example, if you convert decimal 19 to binary, you will compute as remainders 1,1,0,0 and lastly 1. What you should output is the same binary values, but in reverse order: 10011. You should note that a recursive solution allows you to output the results in reverse order naturally. If you use a non-recursive solution you would need to push the values on some type of stack as you compute them and then output the values as you pop them off the stack.

6) Write a recursive function to print a string in reverse order.

7) Write a recursivefunction that returns the index of a given value in an array. If the value is not in the array, return -1. Don't worry about the case of multiple occurrences of the value in the array.

8) Write a recursive function that accepts three arguments: an array, the size of the array and an integer parameter that represents the number of odd values in the array. The function should return the number of odd values in the array or 0 if there is no odd values in the array. There should be no output statement in the function itself.

Here is output produced by my solution (on next page below):

