Problem (1) [15 points]

string FooRecursion (string str) {
    if( str.length() == 0 ) {
        return "";
    }
    char ch = str[str.length()-1];
    string Temp = FooRecursion ( str.substr(0, str.length()-1) );

    return ch+Temp;
}

int main() {
    cout << FooRecursion( "Hello world" ) << endl;
    // cout << FooIterative( "Hello world" ) << endl;
}

(a) What is the output of this program (i.e. the cout in main())?

(b) Write the function FooIterative() that is commented out in main(). This should be a non-recursive function with the same return value.
Problem (2) [15 points] Write a C++ function that takes two ints as inputs: \textit{number}
and \textit{digit}. You may assume the int \textit{digit} is between 0 to 9. This function should return
the number of times the \textit{digit} appears in \textit{number}.

Examples:
\begin{itemize}
  \item number = 1234, digit = 4. Function should return 1.
  \item number = 1004, digit = 0. Function should return 2.
  \item number = 573652853, digit = 5. Function should return 3.
\end{itemize}

Problem (3) [15 points] Write C++ code that reads a sentence and displays the first
and last vowel. You may assume the vowels are: a, e, i, o and u. You can assume there
will be at least one vowel. Hint: some ways of solving this might be easier if you use a
function to help you, though it is not necessary.

Examples:
\begin{itemize}
  \item input = "i like cookies". Should output: "First = i, last = e".
  \item input = "a". Should output: "First = a, last = a".
  \item input = "strings are no fun". Should output: "First = i, last = u".
\end{itemize}
Problem (4) [15 points] Consider the following series sum: \( f(x, N) = x^0 + x^1 + x^2 + \ldots + x^N \). Thus:

\[
\begin{align*}
  f(1, 1) &= 1 + 1 = 2 \\
  f(1, 2) &= 1 + 1 + 1 = 3 \\
  f(2, 2) &= 1 + 2 + 4 = 7 \\
  f(2, 3) &= 1 + 2 + 4 + 8 = 15 \\
  \ldots
\end{align*}
\]

Write a recursive function in C++ that takes two integer inputs \((x\text{ and } N)\) and returns \( f(x, N) \) (i.e. does NOT cout). Assume all inputs will be positive integers.

Hint: You can use pow(a, b) function to calculate \(a^b\). No need to write headers and main function, just write the body of the recursive function.

Problem (5) [15 points] Find 3 possible places for errors in the following code fragment. Explain specifically what causes the error and whether it is a syntax, runtime or logic error. You may assume all includes are done properly and it is using namespace std.

```cpp
const int SIZE = 10; // global var
int[] copyArray(original[SIZE]) {
  int size = original.length();
  int copy[size];

  for(int i = size; i >=0; i--) {
    copy[i] = original[i];
  }

  return copy;
}
```
Problem (6) [15 points] Write a segment of C++ code that reads 10 words from the user and puts them into a file called "magicSpell.txt". The format that you put these words into the file must be:

word1 word1,
word2 word3 word4...
word5 word5,
word6 word7 word8 word7...
word9 word9 word10 word1!

Example input:
cow dog mountain stream squirrel wood metal steel yoyo laptop

magicSpell.txt for example above:
cow cow,
dog mountain stream...
squirrel squirrel,
wood metal steel metal...
yoyo yoyo laptop cow!
Problem (7) [15 points] Write a segment of C++ code (assume somewhere in main()) to do the following. Assume there exists a variable `waterMap` that is a 100 by 100 2D array (grid) of bool values indicating whether or not there is water at that point on the map. Display two things: (1) the number of cells on the map with water (i.e. number of ”true”s in waterMap) and (2) the number of cells on the map that not only contain water but also have a neighbor that contains water. For our purpose, a ”neighbor” is directly next to (up, down, left or right) and not diagonal.

Example 3 by 4 grid:
T T F F
F F T F
T F F T

Sample output for grid above:
5 water cells
2 connected water cells
Problem (8) [15 points] Write a C++ function that takes as input: a sorted integer array, the array size, an index, and a number. This function should return whether the number is between two of the indexes in the array (i.e. not smaller than the first or larger than the last element in the array). If this function returns true, the index passed in should be set to the value $i$ such that: element $i \leq$ number $\leq$ element $i + 1$ (i.e. the smaller index for which the number is between). If there is a tie for possible values of the index, setting it to any valid index is fine.

Sample usage of function in main():
int index;
int number = 5;
bool canSqueeze = search(array, size, index, number)
if(canSqueeze)
{
    cout << "Can fit " << number << " between " << index << " and " << index+1 << " while still being sorted" << endl;
}

Example:
Array = [2, 4, 6, 8], size = 4, number = 3...
Then index = 0 and function returns true
Array = [2, 4, 6, 8], size = 4, number = 7...
Then index = 2 and function returns true
Array = [2, 4, 6, 8], size = 4, number = 20...
Then index is unchanged and function should return false