1014

This takes in the inputs using the scanf functions, then checks how many calculations need to be done.

I used one function from the math.h library to take the radii and apply then to the power of 2. This checks if the difference of the radii is greater than 1, if it isn't it'll just calculate giving the 2 radii. If it is bigger than it will loop for the required amount of times (the radii minus each other -1).

1018

This is a small program that defines reverseWords() as a function. This function takes the current character input, and as long as the current character isn't null, calls the function itself recursively. After calling itself it prints the current character however as this is after the function call it will print them in reverse.

1022

This takes the input string in 1 go and assigns it to a array. The program then loops through each character in the array and checks that characters ascii value against a switch case statement. Whichever value it falls under will increment the required count by 1. The default case is an "other character".

1030

This uses 2 functions from the math.h library. It takes in the numerator, denominator and the decimal point in that order and then calculates the answer of the division. The input numbers are stored in floats as floats are needed as the output. The answer is then multiplied by 10 to the power of the input decimal. This moves the required number to just in front of the decimal point. To retrieve the number from there you take that answer and take mod 10 of that to give the final answer converted to a decimal.

1032

This probably could be a lot more efficient but it's what I got in my head. This has a separate function for calculating the length of the words, this is then used to find the length of the 3 inputs. The program then loops through the input word to find when the input word finds the same letter as the first substring, it then increments both letters by 1 until the end of the substring. If it matches for

the entire substring, it will increment the count of the substring by 1, if not, it will keep scanning through the input for the next iteration of the letter. This then repeats for the 2nd substring (This is where the efficiency could be improved, I believe I could have done this scan as a function and called it twice with the substrings)