Design Document

Design Process

For our design process, we started out with a skeleton code. Just to get an idea of the basic functionality of our code. The three of us created three distinct programs using our own lines of code, and decided to either go with the one that works or to just combine the working or best parts of our program and make one working program. With each line of code and with errors we had to go back, and think of where we went wrong, in our code or in our overall algorithm.

Data Structure

The data structure used in our code are vectors to store the permutations of the digit. Because unlike arrays, vectors are dynamically sized, which is a major advantage. We created 16 vectors, one for each size.
Arrays are used to store the data, organized in for loops. The formal data structure we used for these are arrays.

System Functionality

The program generates permutations for digits 1-16, using elements {1, 2, 3, 4, 5, 6, 7, 8, 9, 0, A, B, C, D, E, F} such that each each permutation of number is divisible by its index+1. the left most digit is evenly divisible by 1, the two left most digits are evenly divisible by 2, the three left most digits are divisibly by 3 and so on… till the last digit.

To generate new permutation the program loops through the integer 1-9, it checks it against the permutation set off the lower order of magnitude/ lower digit number. It converts the each element of the lower order number to string and checks the number for repeating digits. The program then create a new number by multiplying the previous set of digits by 10 and adds the non-repeating number to previous set. It then checks to see if the new number is evenly divisible by the number of digits of the new number; the set of numbers satisfying the condition are added to a new vector. The set in the vector is then used to generate the next.

With vectors created, we begin by going through 1-16 digit numbers, with 1 digit numbers, we run a for loop looping n from 1 to 10 and printing out numbers that are evenly divisible by 1.

With 2 to 9 digit numbers we loop again from 1-9, and cast integer n to a character. We loop through a previous vector and then convert integer vector elements to integer. There is a comparison of a vector element to a character to avoid repetition of numbers using a conditional statement. After a new number is generated, a conditional statement

is used to check to see if the number is evenly divisible by the number of digits. If it is, that number is added to vector, if not it isn't added to vector.

For hex numbers we used string vectors. Use a for loop to iterate through the previous vector. Store elements of previous in a string. Crete a nested for loop to loop through the string. Use insert method to insert hexadecimal number(letter form{a, b, c, d, e ,f}) at the iterator position in the string. Convert the string to a base 16 in using 'stoi' method. Use a conditional to check for even divisibility by 11; if it passes true append the string to the vector next using 'push_back' method. Repeat process: NA to NF.

Team work

Matthew worked on counts 1-4 and planned out the algorithm for the hexadecimals.
Uzair worked on counts 4-9 for counts and the algorithm for hexadecimals.
Reevs worked on counts from counts 4-7 and wrote down the design document.