Prime Cryptarithm

Russ Cox

The constraints of this problem are small enough that we can just try all possible products of 3 digit * 2 digit numbers, and look to see if all the correct digits are used.

The function "isgood" checks that a number is composed only of acceptable digits, and "isgoodprod" checks that all the lines of the multiplication are composed of acceptable digits.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

int isgooddigit[10];    /* isgooddigit[d] is set if d is an acceptable digit
*/

/* check that every decimal digit in "n" is a good digit,
   and that it has the right number "d" of digits. */
int
isgood(int n, int d)
{
    if(n == 0)
                return 0;

    while(n) {
        if(!isgooddigit[n%10])
            return 0;
        n /= 10;
        d--;
    }

    if( d == 0 )
       return 1;
    else
       return 0;
}

/* check that every product line in n * m is an okay number */
int
isgoodprod(int n, int m)
{
    if(!isgood(n,3) || !isgood(m,2) || !isgood(n*m,4))
        return 0;

    while(m) {
        if(!isgood(n*(m%10),3))
            return 0;
        m /= 10;
    }
    return 1;
}

void
main(void)
{
    int i, j, n, nfound;
    FILE *fin, *fout;

    fin = fopen("crypt1.in", "r");
    fout = fopen("crypt1.out", "w");
    assert(fin != NULL && fout != NULL);

    for(i=0; i<10; i++) {
        isgooddigit[i] = 0;
    }
    fscanf(fin, "%d", &n);
    for(i=0; i<n; i++) {
        fscanf(fin, "%d", &j);
        isgooddigit[j] = 1;
    }

   nfound = 0;
   for(i=100; i<1000; i++)
        for(j=10; j<100; j++)
            if(isgoodprod(i, j))
                nfound++;

   fprintf(fout, "%d\n", nfound);
   exit(0);
}
```