Dual Palindromes

Russ Cox

Dual palindromes are actually very common, a fact we can test by writing a program such as this one.

Since they are very common, we can just use a brute force search to test all numbers bigger than s until we find enough dual palindromes.

How do we know they are common enough? Write the brute force program (which is very simple and thus not much effort) and check.

This reasoning is a little circular, but if we had been wrong and ended up needing a more clever and more efficient algorithm, we would have this brute force version to test against.

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

/* is string s a palindrome? */
int
ispal(char *s)
{
    char *t;

    t = s+strlen(s)-1;
    for(t=s+strlen(s)-1; s<t; s++, t--)
        if(*s != *t)
            return 0;

    return 1;
}

/* put the base b representation of n into s: 0 is represented by "" */
char*
numbconv(char *s, int n, int b)
{
    int len;

    if(n == 0) {
        strcpy(s, "");
        return s;
    }

    /* figure out first n-1 digits */
    numbconv(s, n/b, b);

    /* add last digit */
    len = strlen(s);
    s[len] = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"[n%b];
    s[len+1] = '\0';
    return s;
}

/* is number n a dual palindrome? */
int
isdualpal(int n)
```

```
{
    int i, j, npal;
    char s[40];

    npal = 0;
    for(i=2; i<=10; i++)
        if(ispal(numbconv(s, n, i)))
            npal++;

    return npal >= 2;
}

void
main(void)
{
    FILE *fin, *fout;
    int n, s;

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

    fscanf(fin, "%d %d", &n, &s);

    for(s++; n>0; s++) {
        if(isdualpal(s)) {
            fprintf(fout, "%d\n", s);
            n--;
        }
    }

    exit(0);
}
```