USA COMPUTING OLYMPIAD

Greedy Gift Givers

Russ Cox

The hardest part about this problem is dealing with the strings representing people's names.

We keep an array of Person structures that contain their name and how much money they give/get.

The heart of the program is the lookup() function that, given a person's name, returns their Person structure. We add new people with addperson().

Note that we assume names are reasonably short.

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

#define MAXPEOPLE 10
#define NAMELEN 32

typedef struct Person Person;
struct Person {
    char name[NAMELEN];
    int total;
};

Person people[MAXPEOPLE];
int npeople;

void
addperson(char *name)
{
    assert(npeople < MAXPEOPLE);
        strcpy(people[npeople].name, name);
    npeople++;
}

Person*
lookup(char *name)
{
    int i;

    /* look for name in people table */
    for(i=0; i<npeople; i++)
        if(strcmp(name, people[i].name) == 0)
            return &people[i];

    assert(0);  /* should have found name */
}

int
main(void)
{
    char name[NAMELEN];
    FILE *fin, *fout;
    int i, j, np, amt, ng;
    Person *giver, *receiver;
```

```
    fin = fopen("gift1.in", "r");
    fout = fopen("gift1.out", "w");

    fscanf(fin, "%d", &np);
    assert(np <= MAXPEOPLE);

    for(i=0; i<np; i++) {
        fscanf(fin, "%s", name);
        addperson(name);
    }

    /* process gift lines */
    for(i=0; i<np; i++) {
        fscanf(fin, "%s %d %d", name, &amt, &ng);
        giver = lookup(name);

        for(j=0; j<ng; j++) {
            fscanf(fin, "%s", name);
            receiver = lookup(name);
            giver->total -= amt/ng;
            receiver->total += amt/ng;
        }
    }

    /* print gift totals */
    for(i=0; i<np; i++)
        fprintf(fout, "%s %d\n", people[i].name, people[i].total);
    exit (0);
}
```