USA COMPUTING OLYMPIAD

Milking Cows

We read the list of times, sort it by start time, and then walk over the list once, merging overlapping times. Then we walk the list watching for long milking periods and long non-milking periods.

An alternate approach would be to just keep an array of size a million and mark off times. On a nice fast processor, that's probably fast enough, but our above algorithm will work even on slow processors, and it's not much harder to write.

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

#define MAXMILKING 5000

typedef struct Milking  Milking;
struct Milking {
    int begin;
    int end;
};

Milking milking[MAXMILKING];
int nmilking;

int
milkcmp(const void *va, const void *vb)
{
    Milking *a, *b;

    a = (Milking*)va;
    b = (Milking*)vb;

    if(a->begin > b->begin)
        return 1;
    if(a->begin < b->begin)
        return -1;
    return 0;
}

void
main(void)
{
    FILE *fin, *fout;
    int i, j, t, tmilk, tnomilk;
    Milking cur;

    fin = fopen("milk2.in", "r");
    fout = fopen("milk2.out", "w");
    assert(fin != NULL && fout != NULL);

    /* read input, sort */
    fscanf(fin, "%d", &nmilking);
    for(i=0; i<nmilking; i++)
        fscanf(fin, "%d %d", &milking[i].begin, &milking[i].end);

    qsort(milking, nmilking, sizeof(Milking), milkcmp);
```

```
    /* walk over list, looking for long periods of time */
    /* tmilk = longest milking time */
    /* tnomilk = longest non-milking time */
    /* cur = current span of milking time being considered */

    tmilk = 0;
    tnomilk = 0;
    cur = milking[0];
    for(i=1; i<nmilking; i++) {
        if(milking[i].begin > cur.end) {        /* a down time */
            t = milking[i].begin - cur.end;
            if(t > tnomilk)
                tnomilk = t;

            t = cur.end - cur.begin;
            if(t > tmilk)
                tmilk = t;

            cur = milking[i];
        } else {
            if(milking[i].end > cur.end)
                cur.end = milking[i].end;
        }
    }

    /* check final milking period */
    t = cur.end - cur.begin;
    if(t > tmilk)
        tmilk = t;

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

Another Idea (from Jesse Ruderman)

The solution given for milk2 sorts milking periods by start and then walks through them. The solution page also mentions a second possible solution involving a huge array. Here's a third solution that sorts starting and stopping times together, and walks through the "events" of farmers starting and stopping to milk.

```
/* sort the starting and ending times, then go through them from
 start to finish, keeping track of how many farmers are milking
 between each "event" (a single farmer starting and stopping). */

#include <fstream.h>
#include <stdlib.h>

struct event
{
 long seconds;   /* seconds since 5 am */
 signed char ss; /* start = 1, stop = -1 (delta number of farmers milking)
*/
};

int eventcmp (const event *a, const event *b)
{
 if (a->seconds != b->seconds)
  return (a->seconds - b->seconds); /* 300 before 500 */

 return (b->ss - a->ss); /* 1 (start) before -1 (stop) */
}

int main ()
{
 ifstream in;
 ofstream out;
```

```
 in.open("milk2.in");
 out.open("milk2.out");

 int num_intervals, num_events, i;
 event events[5000 * 2];

 in >> num_intervals;
 num_events = num_intervals * 2;
 for (i = 0; i < num_intervals; ++i)
 {
  in >> events[2*i  ].seconds; events[2*i  ].ss = 1;
  in >> events[2*i+1].seconds; events[2*i+1].ss = -1;
 }

 qsort(events, num_events, sizeof(event),
  (int(*)(const void*, const void*)) eventcmp);

/* for (i = 0; i < num_events; ++i)
  out << events[i].seconds
    << (events[i].ss == 1 ? " start" : " stop") << endl; */

 int num_milkers = 0, was_none = 1;
 int longest_nomilk = 0, longest_milk = 0;
 int istart, ilength;

 for (i = 0; i < num_events; ++i)
 {
  num_milkers += events[i].ss;

  if (!num_milkers && !was_none)
  {
   /* there are suddenly no milkers. */
   ilength = (events[i].seconds - istart);
   if (ilength > longest_milk)
    longest_milk = ilength;
   istart = events[i].seconds;
  }
  else if (num_milkers && was_none)
  {
   /* there are suddenly milkers. */
   if (i != 0)
   {
    ilength = (events[i].seconds - istart);
    if (ilength > longest_nomilk)
     longest_nomilk = ilength;
   }
   istart = events[i].seconds;
  }

  was_none = (num_milkers == 0);
 }

 out << longest_milk << " " << longest_nomilk << endl;

 return 0;
}
```