Sample Program #3 (Winning the Lottery) Testing Strategy

The first sample case just tested the "regular" operations of the program on relatively small data. The limitations were that all skip values were 1 or 2, and the winner was a person #1, so this doesn't seem to test situations where all #1s were eliminated before stage 2.

In my first case after the sample, I looked to both minimize and maximize the skip value. I did two lines with a skip of 0 and one line with 97 people had a skip of 96. I manually checked that the first few people removed were what I expected in both cases. By running each group down to 1 person, I expected that all person #1s would be eliminated, providing a better test for phase 2.

In my second case (after the sample), I just changed the threshold to be 40 for each line instead of 1. As expected there was a person number 1 still around for phase 2, but they were in line 7.

The only way to calculate the answer easily is to use a skip of 0. So, I did this for my third case, removing exactly the number of people that I wanted to from each group from the front, so that I could make an interesting tiebreaker case for phase 2.

I repeated this strategy for my next three tests, only using skips of 0 so I could control what was happening in phase 2.

I did one case with one group only, since this is the minimum number of groups. (Other cases already tested the maximum # of groups.)

My next case had the classic Josephus Problem skip = 1 down to 1 person for each line.

My second to last case was a very minimal case with n = 2 and t = 1, so exactly 1 person gets eliminated and I made it a different person in the two lines.

My last case (of lottery01.in) was just sort of thrown in…different line sizes with the same skip (5).

In lottery 2, I stressed tested run times, making lots of lines of the max size. Technically I went over the bounds a bit since I limited each of the two cases to 10^7 number of total skips for a total of no more than 2×10^7, but as long as a program ran in linear time, this wasn't a big deal. Had I caught the issue earlier, I would have just changed the specification to say that sum wouldn't exceed 2×10^7. The second test case in this file is where I tested phase 2 by using a skip of 0 on each line.

In lottery 3, I stress tested removing almost everyone so that phase 2 had a pretty random looking answer.

Just like my other programs, I wrote an alternate solution in Java to double check my C version. In my Java version, to try to be sure I was doing things correctly, I never deleted anything and skipped over "removed people" so to speak. This version was slower, but it added evidence that my C solution was correct, since both versions generated the same answers.