Andrea Donoso

COP 3502C: Arup Guha

Due 10/31/2021

'Scholar.c' Testing Strategy

To ensure that my program is working correctly, I used the following test cases to test whether my program would give me the corresponding outputs, given certain inputs I thought could give issues. The structure of the input is as follows.

c (number of test cases)

n (number of books), L maximum number of pages willing to read

n books' pages

TEST CASE 1:

Tests the overall corresponding results of simple inputs.

Input:

3

3 30

20 5 5

4 40

20 5 5 30

5 50

20 5 5 9 10

Output:

3

3

5

TEST CASE 2:

Tests the maximum number of pages $L \leq 100000000000000 \leq 10^{14}$

Input:

3

2 100000000000000

50000000000000 49999999999999

2 100000000000000

50000000000000 50000000000000

2 100000000000000

50000000000000 50000000000001

Output:

2

2

1

TEST CASE 3:

This test case sets the maximum page limit (L) to greater than (10,000,000,000) = $\mathbf{10^{10}}$

Input:

2

3 10000000000

9999999928 14 58

4 10000000000

3 3000 1500000 9998496998

Output:

3

3

TEST CASE 4:

This test case is generated by a program, of size n = 100000 = $\mathbf{10^5}$

On line 38 of my code, make ‘ **scanf(“%lld”, &books[i]);** ’ a comment.

On line 39 of my code, un-comment ‘ **books[i]=1+rand()%L;** ’.

Perhaps also un-comment the print statement on line 40 to see what random number are chosen as well as line 70 to see what numbers are getting added up to be ≤ L.

Input:

2

100000 10

1 1 1 1 1 1 1 1 1 1

100000 5000

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

Output:

10

358