Data Structure and Advanced Programming

(Programming) Homework #7
Due: 2021/5/4 08:00am (CST)

NOTE: Please upload your C++ source codes (by copy-paste) to PDOGS before the due date and time.

1. (40%) Implement the event-driven simulation of a bank that we talked about in the classroom (actually in the video; you can also refer the Chapter 13.4 in the textbook[1]). A queue of arrival events will represent the line of customers in the bank. Maintain the arrival events and departure events in a priority queue, sorted by the time of the event. Use a link-based implementation for the event list.

   The input is a text file of arrival and transaction times. Your program must count customers and keep track of their cumulative waiting time. These statistics are sufficient to compute the average waiting time after the last event has been processed. Display a trace of the events executed and a summary of the computed statistics (the total number of arrivals and average time spent waiting in line).

   Sample Input / Output
   Each line of the file contains the arrival time and required transaction time for a customer separating by a comma symbol. The arrival times are ordered by increasing time. Please use the sample output format to show the results.

// input
1, 5
 	
// output
Simulation Begins
Processing an arrival event at time: 1
Processing a departure event at time: 6
Simulation Ends
 	
Final Statistics:
 	
Total number of people processed: 1
	// input
	1, 5
	2, 5
	4, 5
	20, 5
	22, 5
	24, 5
	26, 5
	28, 5
	30, 5
	88, 3
	
	// output
	Simulation Begins
	Processing an arrival event at time: 1
	Processing an arrival event at time: 2
	Processing an arrival event at time: 4
	Processing a departure event at time: 6
	Processing a departure event at time: 11
	Processing a departure event at time:

[1] *Data Abstraction and Problem Solving with C++: Walls and Mirrors* by Carrano and Henry, sixth edition, Pearson, 2012.

Average amount of time spent waiting: 0	16 Processing an arrival event at time: 20 Processing an arrival event at time: 22 Processing an arrival event at time: 24 Processing a departure event at time: 25 Processing an arrival event at time: 26 Processing an arrival event at time: 28 Processing an arrival event at time: 30 Processing a departure event at time: 30 Processing a departure event at time: 35 Processing a departure event at time: 40 Processing a departure event at time: 45 Processing a departure event at time: 50 Processing an arrival event at time: 88 Processing a departure event at time: 91 Simulation Ends Final Statistics: Total number of people processed: 10 Average amount of time spent waiting: 5.6

2. (30%) Don has n hammers, where $1 <= n <= 105$. The hammer i has attack power P_i, where $0 <= P_i <= 1014$. He needs to destroy a wooden door, the strength of which is d, *where* $1 <= d <= 1017$. When a hammer with power P_i strikes the door, the strength of the door is decreased by P_i (if $d < P_i$, then d becomes 0) and the attack power of the hammer becomes $floor(P_i / 2)$. Don has k ($1 <= k <= 105$) trials, and he has to break the doors within those k trials. If it is not possible, print -1. If possible, print the least number of hits required.

   Sample Input / Output
   The first input line contains 3 integers, n, d, k, and the second line contains n space separated positive integers, denoting the powers of the n hammers. Print the number of attacks required (less than or equal to k), or -1 if it is not possible to destroy the door within k trials.

// input 3 30 4 10 5 11	// input 5 40 4 10 5 11 3 8

// output 4	// output -1

3. (30%) Mike has an integer array *A starting with index 1*. For each index *i*, he wants to find the product of the largest, second largest and the third largest integer in the range [1, *i*]. Note: Two numbers can be the same value-wise but they should be distinct index-wise.

   Sample Input / Output
   The first input line contains an integer N >= 3, denoting the number of elements in the array A. The next line contains N space separated integers, each denoting the i-th integer of the array A. Print the answer for each index i in each line, but ignore the answer when i < 3.

// input
5
1 2 3 4 5
 	
// output
6
24
60
	// input
	6
	2 1 3 6 5 4
	
	// output
	6
	36
	90
	120