Entrance Evaluation

Objective
Give practice with sorting in C.
Give practice with Queues in C.

Story
Your park is extremely popular. It's too popular. You have many individuals complaining about the wait time to enter the park. You want to analyze the time taken to enter the park.

You set up an expensive system to track the time at which groups of park patrons arrived, the size of each group, and the time each group took to purchase their tickets once at the front of the line. However, the system stored the data in a strange order, and you need to write a program to analyze the data.

There were only 2 ticket counters (left and right) at which to purchase tickets. Both ticket counters had their own line. You know that groups that arrive are good at counting and will always enter the line that has the fewest people. If the 2 lines have the same number of people, then the group enters the "left" line. Once a group entered a line, the group stayed in that line, even if the other line emptied. Once a group is at the front of the line, they begin working on purchasing the tickets, and the time taken is based on given time for that group. If a group is finished being processed at the same time as another group arrives, then the arrival happens first.

Your task is to find the sum wait times of each person (not just each group). No two groups arrived at the same time.

Problem
Given a list groups with their sizes, their arrival time, and the time to be processed once at the front of the line.

Input
Input will begin with a line containing 1 integer, n ($1 \leq n \leq 500{,}000$), representing the number of groups to process. The following n lines will each contain 3 integers: s, a, and p ($1 \leq s \leq 1{,}000{,}000$; $1 \leq a \leq 1{,}000{,}000{,}000$; $1 \leq p \leq 1{,}000{,}000$) representing the group size, arrival time, and processing time respectively.

Output
Output should contain 1 integer representing the sum time taken by all people that waited in line.

Sample Input	Sample Output
4 1 4 10 1 2 10 1 1 10 1 3 10	56
3 2 10 10 3 9 1 1 1 20	65

Explanation

Case 1

There are 4 groups. The groups all have 1 person in them. They all take 10 units of time to be processed.

The first arriving group is actually the third group in the input. They arrive at time 1. They take 10 units to process. They would finish at time 11. They enter the left line. They took 11 - 1 total time.

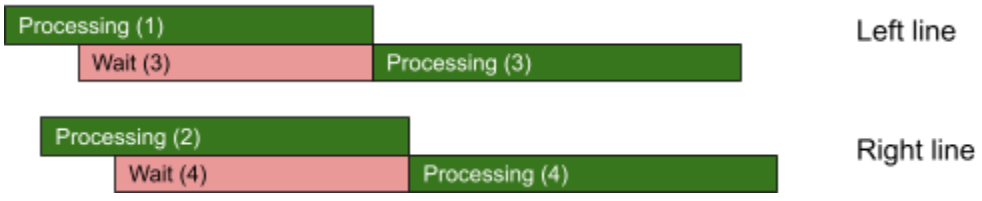
The second arriving group is the second group in the input. They arrive at time 2. They take 10 units to process. They would finish at time 12. They enter the right line, since the left line has the first arriving group. They took 12 - 2 total time.

The third group to arrive is the fourth group in the input. They arrive at time 3. They take 10 units to process, but they won't be processed until time 11, since 1 is being processed in their line. They finish their process at time 21. They took 21 - 3 total time.

The fourth group to arrive is the first group in the input. They arrive at time 4. They take 10 units to process, but they won't be processed until time 12, since 2 is being processed in their line. They finish their process at time 22. They took 22 - 4 total time.

The complete wait time is therefore 56 (10 + 10 + 18 + 18).

Below shows pictorially how the groups wait or get processed based on their arrival times.



Case 2
There are 3 groups.

The first group to arrive is the third group in the input. They arrive at time 1. They take 20 units of time to be processed. They will finish at time 21. They have 1 person in their group. The total time they took was 21 - 1 time.

The second group to arrive is the second group in input. They have 3 people in their group. They enter the right queue. They take 1 unit of time to be processed. They enter at time 9 and finish at time 10. They have 3 people in their group. The total time they took was 10 - 9 (time taken) times 3 (for each person).

The third and last group to arrive is the first group in the input. They have 2 people in their group. They arrive at time 10 and take 10 time units to finish being processed. At that moment there are still 3 people in the right queue, so they enter the left queue, which only has 1 person. They end up waiting until time 21 to be processed. This means that they finish being processed 10 units later at time 31. They contributed 31 - 10 (time taken) times 2 (for each person).

The complete wait time is therefore 65 (20 + 3 + 42)

Hints

NO int-ing: Use the data type long long int to store your answer instead, because the data type int does not have enough precision.

Simulate: You need to simulate the groups entering and being processed.

Sorting by Arrival: You should sort by arrival time to ensure that the groups that arrive later don't affect groups that arrive earlier. You should allow groups to "enter" the line using a loop over all the groups after sorting by arrival time.

Simulating the Line: You should use the First In First Out (FIFO) data structure (Queue) to ensure the groups are processed in the order they arrive.

Keep track of when you started processing the front group of each line.

When a line is empty and a group is added to it, then update that the group has started its processing at that moment.

The time a group contributes is the time they waited times the group size.

2 Queues: You should have a queue for both the left and the right line.

Update on Arrival: When a group "enters" (you are trying to add them to the queue) first remove groups from the front of each line prior to determining which line the new group goes into.

Store People in Line: Store for each queue the number of people in each line rather than looping through the line to find the size.

Clear the Lines: After all the groups have been added the queue should be processed until emptied.

Pseudo Code: Below is the pseudo code for the simulation

```
Create empty loops
Loop through the groups in order of arrival time
   The newest time will become the arrival time of this group
   While a the front of a group can be processed
      Remove the front of the queues (update the answer)
   Determine which line the group will enter
      Add the group to the "best line"
      Update the processing time if the queue was emptied
While the queues are not empty
   Process the first group of the queue
   Remove the front of the queue (update the answer)
```

Instructor Structs and Prototypes: Below are recommended structs and function prototypes.

```
struct Group {
   long long int arrive, size, process;
};
struct Node {
   Group data;
   Node * next;
};

// Sort an array of groups in increasing order of arrives
void sort(Group ** array, int size);

// Create a node with a new group as data
// The node is for a circular linked list
Node * newNode(Group newGroup);

// Add a group to the tail of a circular linked list
// Return the resulting tail
Node * addTail(Node * tail, Group newGroup);

// Remove the head node of a circular linked list
// Return the resulting tail
Node * removeHead(Node * tail, Group newGroup);
```

Grading Criteria

- Good comments, whitespace, and variable names
    - 15 points
- No extra input output (e.g. input prompts, “Please enter the number of friends:”)
    - 10 points
- Store the groups in some structure
    - 5 points
- Write a custom sort to sort the groups by arrival time
    - 10 points
- Ensure that groups are processed in order of arrival (loop and queue)
    - 10 points
- Programs will be tested on 10 cases
    - 5 points each

No points will be awarded to programs that do not compile using “gcc -std=gnu11 -lm”.

Sometime a requested technique will be given, and solutions without the requested technique will have their maximum points total reduced. For this problem use a sort and queue. ***Without this programs will earn at most 50 points!***

Any case that causes a program to return a non-zero return code will be treated as wrong. Additionally, any case that takes longer than the maximum allowed time (the max of {5 times my solutions time, 10 seconds}) will also be treated as wrong.

No partial credit will be awarded for an incorrect case.