Objective: Determine exercise or not to maximize the total performance

- Brute force: $\Omega\left(2^N\right)$. There are two options for each day: exercise or not.
- The problem is linearly ordered → dynamic programming?

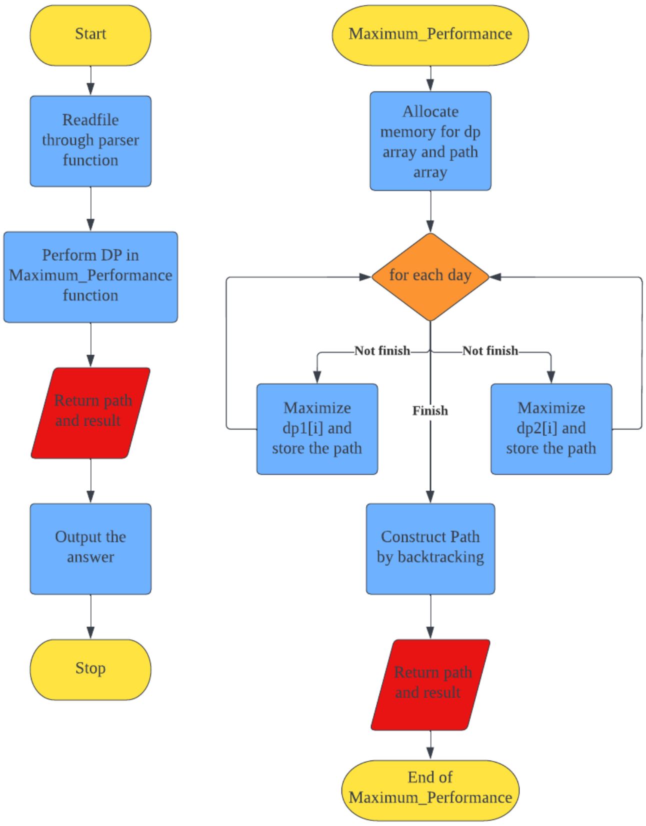
dynamic programming to solve the problem:

define dp1[i-1]: maximum performance if the ith day exercise
define dp2[i-1]: maximum performance if the ith day rest
(because the index starts from 0)

Base case: The first-day exercise or not. (Initial performance set to 0)
dp1[0] = 0 + A - 1 * 1 * B
dp2[0] = 0 - R[0]

Transition function:
dp1[i] = max(dp1[i-1] + A - X * X * B, dp2[i-1] + A - B);
dp2[i] = max(dp1[i-1] - R[i], dp2[i-1] - R[i]);

Flow chart:


Start
Readfile through parser function
Perform DP in Maximum_Performance function
Return path and result
Output the answer
Stop
Maximum_Performance
Allocate memory for dp array and path array
for each day
Not finish
Not finish
Finish
Maximize dp1[i] and store the path
Maximize dp2[i] and store the path
Construct Path by backtracking
Return path and result
End of Maximum_Performance

Time complexity analysis:

Read file: $\theta(N)$

Bottom-up dynamic programming for memorization: $\theta(N)$

Backtracking the solution path: $\theta(N)$

Output answer: $\theta(N)$

Therefore, the overall time complexity is $\theta(N)$