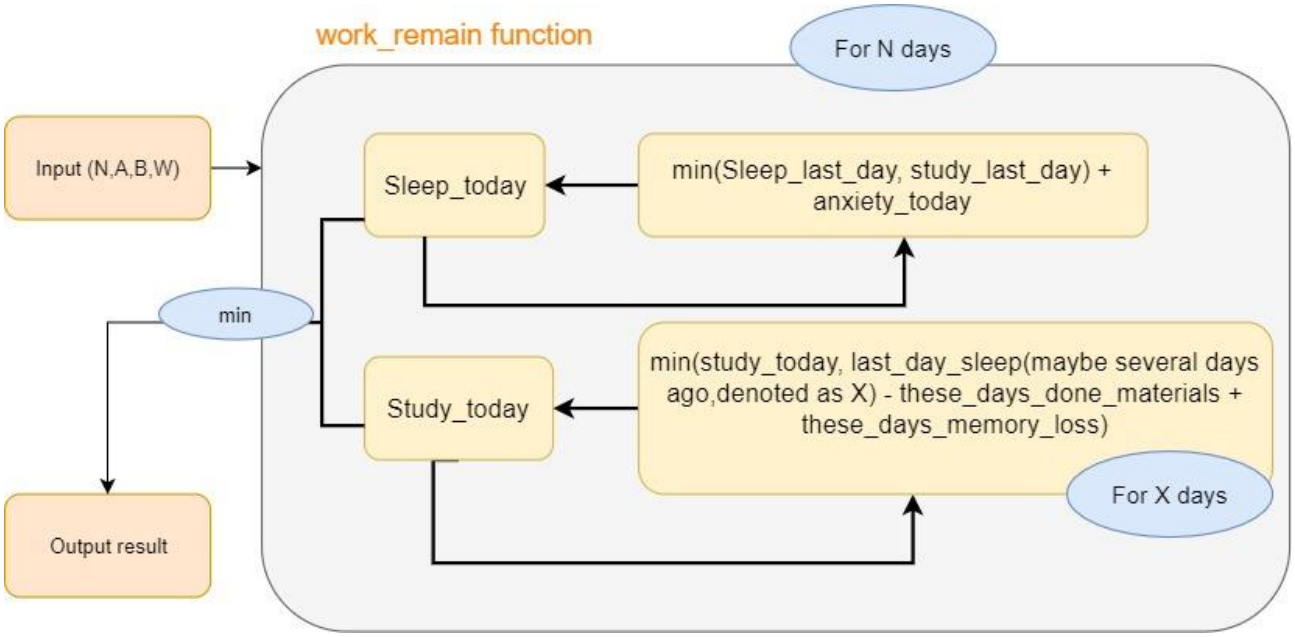
Program assignment 3

Flowchart:



Pseudocode:

```
Work_remain (N,A.B.W.D)
    dp[N+1][N+1] = {0}
    dp[0][0] = W, dp[0][1] = W
    for i from 1 to N
        dp[i][0] = min(dp[i-1][0], dp[i-1][1]) + D[i]
        for X from 1 to i
            dp[i][1] = min(dp[i-1][1], dp[i-X][0] – A*X + B_for_X_days)

    return min(dp[N][0], dp[N][1])
```

Method:

I use dynamic programming to implement this function.
Since there are 2 conditions: sleep or study, I built a table in size N*2,which N is for total days and 2 is for sleep / study. The transition function needs to transfer on each day until last day to get the best working remaining. If we sleep, we only need to add an anxiety factor, so we can get the sleep transition function easily by:
today-sleep = take the minimum of remaining workload of last-day-sleep result and last-day-study result, then add today's sleeping punishment.

But today-study is much more hard to think of. We need to know how many days we've studied so far to get the memory loss X for addition. But how can we get the X?
For example, if I study for 5 days in a row, which means I must slept on 6^{th} day before. I should get the 6^{th} day best sleep result to decide how many days I should continuously study on these 5 days. So I run each cases one by one, if I study for 1,2,3..….i(today) days in a row(in variable X), do the calculation of workload minus and memory loss addition for each cases and continuously compare the results with today-study one to get the minimum result of study so far, which is:
today-study = take the minimum of remaining workload of today-study result and last-day-slept-until-now result.
Keeping updating the best today-study result so far(X++) then goes on next day (i++).
I set the first self-comparing-element as INT_MAX to prevent the comparison going wrong.
Finally, the result will be the minimum of last-day-sleep result and last-day-study result.

Time Complexity:

Two for loops :
(1~N) → each day of sleep result and study result.
(1~i) → from first day to today, best study days distribution.
total time complexity O(N^2)
I know there's a O(N) method, but I'm too stupid to think of. ☹