• What is the problem? (concise description/definition, no story)
m players
n items on a line, each with value s(0)...s(n-1)

players take away rightmost or leftmost item, in turns, beginning with player 0.

Calculate how much value player k can gain maximally.
Make no assumption about other players, they do not have to individually maximize their gains. They could be trying to minimize the gains of player k.

• How do you model it? (What mathematical concepts do you use to formally describe and reason about (and eventually solve) the problem?)
We get a min-max problem
Player k is maximizing in his roughly n/m thurns
The others are minimizing k's gains in the other turns

We have different cases:
k never gets to play -> gains=0
k at least one turn
- there is a pre-phase and a post-phase
- if k has more than one turn there is at least one mid-phase

For mid-phase, d>=m, t-th turn for k:
assume s[i]...s[i+d-1] are left to take:
val[i,t] = max(
          s[i] + min_{j=1...m} (val[i+j,t+1]),
          min_{j=0...m-1} (val[i+j,t+1]) + s[i+d-1]
)
so take either left or right and let others minimize over m-1 turns.

For begin phase:
minimize over first k turns of the others

For last phase:
Take only left or right most element, the others will be taken by the other players

• How do you algorithmically solve it? (Describe the algorithm, first in a rough overview, and then selectively in more detail. Analyze the runtime.)
DP, bottum up.
Start by analyzing how many turns k gets.
Start with post-phase, calculate for all possible i.
Do same, turn by turn, for mid-phases, each for all possible i
Finally, do pre-phase minimization.
Each phase relies on results of previous phase.

There are O(n/phases)
Each phase has to calculate the value for each i -> O(n)
Each value is a minimization over O(m) values
Thus we get a total runtime of O(n/m n m) = O(n^2)
This is acceptable, given size of n

• How do you implement and test it? (Explain the most interesting steps relevant for your implementation. Only here you may refer to the printout of the code you brought.)
Implementation:
via one table array of size n.
new phases can overwrite table for old phases, if go over i increasingly
This improves performance (cache)
Further minor optimizations

Test via test cases
print out what pre-mid-post turns it does (then validate manually for some cases)
gdp step through impl, validate indices

• Overall: What challenges did you face in these different phases and how did you overcome them? What was most difficult for you during the process?

I came up with the phases idea quickly, maybe by chance.
Most difficulty I had with calculating the correct phase-numbers
Further there were some index issues.
Next time I should think harder about them before just coding.