Dynamic Programming

- Recurrence
- ... w/ overlapping subproblems
- ⇒ save results for subproblems in a data structure.

- Each week: 1 hi-stress 1 lo-stress job.
- Pay different amts.
- To do hi-stress job, must take previous week off.
- Goal: maximize profit.

eg.

Week:	1	2	3	4	5	6
Lo:	6	$\boxed{12}$	~~9~~	30	17	
High:	$\boxed{10}$	5	18	80	$\boxed{1000}$	...?

Greedy: take higher-paying job each week, remove prev week if it's high-stress.

Doesn't really work.

key idea: add a parameter. Let $M(k)$ = max \$ we can make working only weeks 1..k. We can come up w/ a recurrence for M.

- $M(0) = 0.$
- $M(1) = \max(L_1, H_1)$
- $M(k) = \boxed{\max} \begin{cases} L_k + M(k-1) \\ H_k + M(k-2) \end{cases}$

L_i = lo-stress job \$ wk. i

H_i = hi- " " " "

— take lo-stress wk k, do our best in prev. wks.

— take hi-stress wk k, take k-1 off, do our best in 1..k-2.

Clearly, calls to M will overlap. Solution: Store results in a table. (Memoization)

Week:	1	2	3	4	5
Lo:	6	12	9	30	2
High:	10 +	5	18	80	6
M:	0	10	22	31	104
J:	H	L	L	H	L

1	2	3	4	5
H	L	X	H	L

Note we are just filling in a length-(n+1) array, each entry took $\Theta(1)$ to compute, so whole thing is $\Theta(n)$.

This doesn't remember which choice we should make @ each week. Can make another table of booleans:
$J[k]$ = should we take hi-stress job @ week k to maximize earnings for weeks $1..k$.