**Qn Link :** <https://www.desiqna.in/10567/barclays-sde-coding-oa-questions-and-solutions-set-8-2022-dp>

**Question Summary :**

* Given an 2D array consists of easy task and hard task
* Each task is associated with sum salary.hard task have high salary.
* If you do hard task then yu can’t do any task in the previous day.
* Maximize the salary got by stephen.

**Observation :**

* We have three possibilities ,
  + Do easy task
  + Do hard task
  + Do nothing
* If we do easy task , then we need to take the maximum of sum of easy task of previous day + hard task + nothing
* If we do hard task then , we only take what we ve done till previous day nothing.
* For nothing task , we will calc max of easy task , hard task , nothing task on previous day.

**Recurrence Relation :**

* ,dp[i][easy] = max ( dp[i - 1][easy] , dp[i - 1][hard] , dp[i- 1][nothing]) + x
* , dp[i][hard] = dp[i - 1][nothing] + Y
* , dp[i] [nothing] = max ( dp[i - 1][easy] , dp[i - 1][hard] , dp[i- 1][nothing])

**Code :**

class Solution {

private int max(int a , int b , int c){

if(a > b && a > c){

return a;

}

if(b > c){

return b;

}

return c;

}

public int maxMoney(int n , int [] easy , int [] hard){

int [][] dp = new int[n + 1] [4];

dp[1][1] = easy[0];

dp[1][2] = hard[0];

dp[1][3] = 0;

for(int i = 1 ; i < n ; i++){

dp[i][1] = Math.max(dp[i - 1][1] , dp[i - 1][2] , dp[i - 1][3]) + easy[i];

dp[i][2] = dp[i - 1][3] + hard[i];

dp[i][3] = Math.max(dp[i - 1][1] , dp[i - 1][2], dp[i - 2][3]) + 0;

}

return max(dp[n][1] , dp[n][2] , dp[n][3]);

}

}