	10	10	6	/2	0	2	3
~	Married Print		-	-	-		_

Partition equal subset sum.

i/p→ {1,5,11,5} O/p→ True

1+5+11+5=22

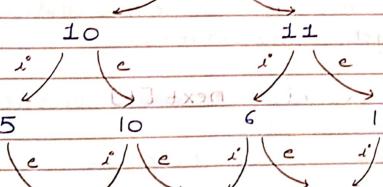
sym = 11

2) - dividing by 2 as equal subset sum.

Now we need to find the subset having

Include-exclude Pattern

{1,5,11,5}, target = 11



5 10 X 6 0 11 Retwin tyme

<u>5</u>

Base case - Retwin true

In main function, simply calculate the sum of elements of array. We have to add one case

Scarnieu with CdM

(Base case)

~	
	where if sum is odd , then we have to
	simply return false Initialize Index = 0,
	target = Sum /2 ;
	1/ Recursive code
	bool solve Rec (int index, vector (int) frums,
	int target) {
	if (index > = nums size())
	retwin 0;
	if (target <0)
	return Oj
	if (target ==0)
	retwin 1)
	L D (in law hat a number
	bool include = solve Rec (index+1, nums)
	target - nums [index]);
	bool exclude = Solve Rec (index+1)
	nums, target)
	return include ll exclude
	3
	10 10 10 10 10 10 10 10 10 10 10 10 10 1
	1/ Top-down approach
	bool solve Top Down (int index, vector (int)
	2 nums , int target, vector < vector < Int
	2db){
	//Base case
	if (index >= nums size ()) {
	return o;
	if (target < 0)
	retwin O;
<b>C</b>	Scarnieu with Ca

Scarined with Am



```
if (target ==0)
           return 1;
     //Step 3: Check if answer already exists
    if (dp [index] [target] | = -1)
             return of [index][target]
    Minclude call
   bool include = Solve Top Down (index +1, nums,
   target - nums [index], dp);
   Mexclude call
   bool exclude = solve Top Down (index +1, nums,
  target, db);
  // Step 2: Store answer in dp array.
  dp[index][tagget] = include | lexclude;
 return dp [index] (target];
//Bottom up approach
bool solve Tab (vector <int>& nums, int target)
     int n = nums·size();
   //Step 1: Create of array
   Vector < vector < int >> db (nums.size()+1,
   vector <int>(target +1,0));
  // Step 2: Observe base case of top down
  for (int i=0 > i<nume size() > i++) {
         db[i][0]=1;
1/Step 3: Reverse flow of top-down
for (int index = n-1; index >=0; index--) {
 for (int t = 1; t < = target ; t++){
       bool include = 0;
       if (t-nums [index] >=0)//Valid index check
            include = dp[index+1](t-nums [index]];
```

Scarnieu with cam

	bool exclude = dp[index +1] [t];
	ap [index] [t] = include   lexclude;
	4
	3
	// Ans is at ab [o][target] as we are going to 0.
	// Ans is at dp [o][target] as we are going to 0.  Yetwin dp [o][target];
	3
	yt a little to the second of t
Note	In top-down approach, simply create
	2D vector in main () as
	vector <vector <int="">&gt; dp (nums size ())</vector>
	vector <int>(tanget +1, -1));</int>
	Maria Cara Language Cara Cara Cara Cara Cara Cara Cara Car
$\square$	Why 2D away is used? As in the recursive call, 2 variables
	are changing i.e index and target.
$\bigcirc$	
<u>\\\</u>	Space optimization possible or not?
	Yes space optimization is possible here, delindex][t] depends on include and
	exclude.
	PXCIQQC.
	include + dp [index +1][t-nums [index]]
	exclude - db [index +1][t]
	Hence dp[index][t] depends upon the next
	100.
	Make 2 1D arrays.
	vector <int> cour (target +1,0);</int>
	Make 2 ID arrays.  vector <int> cwr (target +1,0);  vector <int> next (target +1,0);</int></int>
- 11	
	Simply replace of [index] with our and
	\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
liv s	Scarnieu with cam

	dp[index +1] as next in the bottom up
	approach.
	Also we need to do shifting i.e next = cwu as
	we are going upwards.
Qa	Number of dice rolls with target sum.
	Here the pattern that we will be applying -
	is distinct ways in which for loop will be
	used.
	o dices
	i/b → n = 2, k = 6, target = 7 → faces
	In this test case we will have 36 choices
	or paus.
	{1,13 {2,13 {3,13 {4,13 {5,13 {6,13}
	(
	$\{1,3\}$ $\{2,3\}$ $\{3,3\}$ $\{4,3\}$ $\{5,3\}$ $\{6,4\}$ $\{1,4\}$ $\{2,4\}$ $\{2,4\}$ $\{6,4\}$
	$\{1,5\}$ $\{2,5\}$ $\{3,5\}$ $\{4,5\}$ $\{5,5\}$ $\{6,5\}$
	{1,63} {2,63} {3,63} {4,63} {5,63} {6,6}
	(1) 6)
	Total ways = 6 in which we can get sum = 7.
	- True volgs - III.
	Code
_	-555
_	long long int MOD=1000000007; given in question-
	int solve Rec (int no int ko, int tanget) {
	11 Base case
	if $(n < 0)$
	retuin 0;
1	H

Scanned with Cam

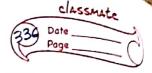
	if (n = = 0 4& tanget = = 0)
	retwin 1;
	if (n == 0 && target != 0)
	retunn O
	if (n!=0 && tanget ==0)
	retwin 0;
	long long int ans = 0;
	long long int ans = 0; for (int i = 1; i < = k; i++) { > tanget reduced}
	ans += solve Rec (n-1, R, target-i);
	3 Lice reduced
	return ans ;
	3
	1/ Top down approach
	long long int solve Top Down (int n, int k,
	Int target, vector < vector < long long int >>4
	(ap) 2
	// Base case
	if (n < 0)
-	return 0;
	if (n = = 0 && target = = 0)
	if (n = = 0 & & tanget 1 = 0)
	return 0;
	if (n !=0 & & target ==0)
	Yeturn O;
	// Step 3: Check if ans already exists  if (db(n)[tanget]  = -1)
	if (dp(n)[tanget] 1 = -1)
	return op(n)[target];
	long long int ans = 0; for (int i=1) i <= k; i++) {
	$\begin{cases} 1 & \text{long long in } 1 < \text{long long in } 1 < long long long long long long long long $
	long long int recAns = 0;  if (target - i >=0) //valid index checks
	(Wigh 2 1-0) // valid index check

Scarineu with Jam



	recAns = solve Top Down (n-1, k, target-i, db);	
	3	
	recAns = recAns 1/2 MOD;	
	ans = ans % MOD;	
	ans = (ans + secAns) % MOD;	
	3	
	// Step 2: Store in dp averay	
	dp[n][target] = ans;	
	return db [n] [target];	
	3	
	and the state of t	
Mof	We have created of array in main () as	
	vector < vector < long long int >>db (n+1, vector < long long int > (target +1, -1));	
	vector < long long int> (target +1, -1));	
	Why 2D DP is used here?	
	2D DP was used here as target and no	
	2 variables are changing.	
	1/ Roll o	
	Bottom up approach	
	int solve Tab (int n, int k, int target) {	
	Vector < vector < long long int >> db (n+1)	
	Vector evector exoring xong the prost + 1.0));	
	Vector < long long int>(tanget+1,0)); //Step 2: Observe base case of top down	
_	dp[0][0] = 1;	
	// Step 3: Reverse flow of top down	
-	for (int index = 1 ; index <= n; index ++) {	
	for (int $t = 1$ ) $t < = tanget$ ; $t + +$ ) {	
	long long int on = 0	
	long long int ans = 0; For (int $i=1$ ; $i < = R$ ; $i++$ ) {	
	long long int yecAns = 0;	
	Aung Kung Ins	

Scarined with Cam



	if (t-x>=0)
	recAns = dp (index - 1) [t-i];
	ans = (ans 1. MOD + HecAns 1. MOD) 1. MOD;
	3
	dp [index][t] = ans i
	2
	retwin dp[n][target];
	J Company of the state of the s
Vote	Anglanda
AULE 1	Instead of n, we used index and instead of
	target, we used t in bottom up code.
!	Share obtingination in Langill
	Space optimization is possible or not?
	de [index] is dependent upon de [index-1] i.e previous row & hence space can be
4	optimized.
	Create 2 1D arrays
,	vector <int> prev (target+1,0); vector <int> curr (target+1,0);</int></int>
1	vector (int) (wur (tanget +1 n).
<u></u>	Now instead of dp [o][o] use prev [o].  dp[index] - use curve.  lb [index - 1] - use prev
	ap[index] - use cury.
d	b [index -17 -> use prev.
3) 5	Shifting logic will he business
	Shifting logic will be prev = cur as ve are going downwards.
	A seed of the seed

Scarnieu with cam