Dynammic Programming

1. ## Playing with Numbers

Program:

```c
#include <stdio.h>
#include <strings.h>

long int function(long int dp[], int n){
    if(dp[n]!=0){
        return dp[n];
    }
    if(n <= 2){
        return 1;
    }
    else{
        dp[n]=function(dp,n-1)+function(dp,n-3);
    }
    return dp[n];
}
int main(){
    int n;
    scanf("%d", &n);
    long int dp[n+1];
    bzero(dp,(n+1)*sizeof(long int));
    long int ans=function(dp,n);
    printf("%ld", ans);
    return 0;

}
```

Output:

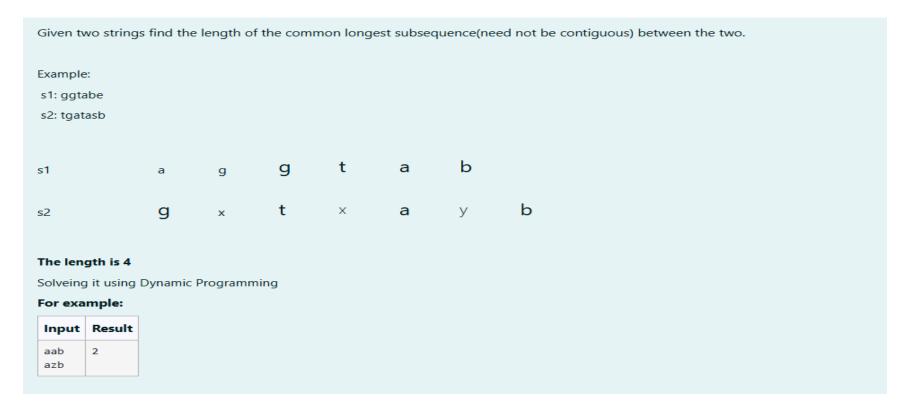|	Input	Expected	Got	
✔	6	6	6	✔
✔	25	8641	8641	✔
✔	100	24382819596721629	24382819596721629	✔

Passed all tests! ✔

2. Playing with chessboard

Playing with Chessboard:

Ram is given with an n*n chessboard with each cell with a monetary value. Ram stands at the (0,0), that the position of the top left white rook. He is been given a task to reach the bottom right black rook position (n-1, n-1) constrained that he needs to reach the position by traveling the maximum monetary path under the condition that he can only travel one step right or one step down the board. Help ram to achieve it by providing an efficient DP algorithm.

Example:

Input

3
1 2 4
2 3 4
8 7 1

Output:

19

Explanation:

Totally there will be 6 paths among that the optimal is
Optimal path value:1+2+8+7+1=19

Input Format

First Line contains the integer n
The next n lines contain the n*n chessboard values

Output Format

Print Maximum monetary value of the path

Program:

```c
#include <stdio.h>

int maximum(int a, int b) {
    return (a > b) ? a : b;
}

int chess(int input[][100], int row, int col, int n) {
    if (row == n - 1 && col == n - 1) {
        return input[row][col];
    }
    int right_sum = 0, down_sum = 0;
    if (col + 1 < n) {
        right_sum = chess(input, row, col + 1, n);
    }
    if (row + 1 < n) {
        down_sum = chess(input, row + 1, col, n);
    }
    return input[row][col] + maximum(right_sum, down_sum);
}

int main() {
    int n;
    scanf("%d", &n);
    int input[100][100];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &input[i][j]);
        }
    }
    int ans = chess(input, 0, 0, n);
    printf("%d\n", ans);
    return 0;
}
```

Output:

	Input	Expected	Got	
✔	3 1 2 4 2 3 4 8 7 1	19	19	✔
✔	3 1 3 1 1 5 1 4 2 1	12	12	✔
✔	4 1 1 3 4 1 5 7 8 2 3 4 6 1 6 9 0	28	28	✔

Passed all tests! ✔

3. Longest Common Subsequence

Given two strings find the length of the common longest subsequence(need not be contiguous) between the two.

Example:
s1: ggtabe
s2: tgatasb

s1		a	g	g	t	a	b
s2		g	x	t	x	a	y	b

The length is 4

Solveing it using Dynamic Programming

For example:

Input	Result
aab azb	2

Program:

```c
#include<stdio.h>
#include<strings.h>
#include<string.h>
int max(int a,int b){
    return a>b?a:b;
}

int lcs(int cost[100][100],char* s1,char*s2,int i,int j){
    if(i==0 || j==0) return 0;
    if(cost[i][j]!=-1) return cost[i][j];
    if(s1[i-1]==s2[j-1]){
        cost[i][j]=lcs(cost,s1, s2, i - 1, j - 1) +1;
    }
    else{
        cost[i][j] = max(lcs(cost,s1, s2, i - 1, j), lcs(cost,s1, s2, i, j - 1));
    }
    return cost[i][j];
}

int main(){
    char s1[100], s2[100];

    scanf("%s %s", s1, s2);

    int len1 = strlen(s1);
    int len2 = strlen(s2);
    int cost[100][100];
    memset(cost,-1,sizeof(cost));

    printf("%d\n", lcs(cost,s1, s2, len1, len2));

    return 0;
}
```

Output:

	Input	Expected	Got	
✔	aab azb	2	2	✔
✔	ABCD ABCD	4	4	✔

Passed all tests! ✔

4. Longest non-decreasing Subsequence

Problem statement:

Find the length of the Longest Non-decreasing Subsequence in a given Sequence.

Eg:

Input:9

Sequence:[-1,3,4,5,2,2,2,2,3]

the subsequence is [-1,2,2,2,2,3]

Output:6

Program:

```c
#include<stdio.h>
#include<string.h>
int max(int a,int b){
    return a>b?a:b;
}
int lns(int * arr,int n){
    int cost[n];
    for (int i = 0; i < n; i++) {
        cost[i] = 1;
    }
    for (int i = 1; i < n; i++) {
        for (int j = 0; j < i; j++) {
            if (arr[j] <= arr[i]) {
                cost[i] = max(cost[j] + 1, cost[i]);
            }
        }
    }
    return cost[n-1];
}
int main(){
    int n;
    scanf("%d",&n);
    int arr[n];
    for(int i=0;i<n;i++) scanf("%d",&arr[i]);

    printf("%d",lns(arr,n));
    return 0;
}
```

Output:

	Input	Expected	Got	
✔	9 -1 3 4 5 2 2 2 2 3	6	6	✔
✔	7 1 2 2 4 5 7 6	6	6	✔

Passed all tests! ✔