Problem:

In this exam, you are given two 2D boxes consisting of full and empty cells. The goal is to convert the first box into the second with the minimum cost of operations (the operations are defined below). The boxes are represented as 2D char arrays. In the arguments, both boxes will be defined to include the same number of rows, yet the number of their columns may be different. For instance, Box-1 can be an array of the size of 10x12 whereas Box-2 can be an array of the size of 10x15. In order to represent empty cells, '-' character is used and for the full cells a letter is used. In the figure below, a few input box illustrations are given:

(a)

BOX-1

M
R
V
-

A box defined as an array of 4x1

BOX-2

-	M	V
M	-	M
R	-	R
V	R	-

A box defined as an array of 4x3

Resulting mem

0	3	5	8
3	4	6	8

(b)

BOX-1

-	B	C
B	-	C

A box defined as an array of 2x3

BOX-2

A	-	C	A	B
-	C	C	A	-

A box defined as an array of 2x5

Resulting mem

0	1	2	4	6	7
1	2	2	4	6	7
2	2	3	5	7	6
4	4	4	3	5	6

(c)

BOX-1

A	A	C	B	-
B	C	-	-	A
A	C	-	A	-
-	C	-	-	C

A box defined as an array of 4x5

BOX-2

A	C	B	-	A	B	A
B	A	B	-	C	-	C
B	-	-	A	C	A	B
C	-	B	-	B	A	B

A box defined as an array of 4x7

Resulting mem

0	4	6	9	10	14	17	21
3	3	5	8	9	13	16	20
7	5	7	10	11	10	13	17
8	6	7	10	11	11	14	18
10	8	9	12	12	13	13	17
12	10	10	13	14	15	15	19

(d)

BOX-1

-	Y	W	Y	-	-
-	Z	Y	Y	-	W
W	Y	Z	Y	-	-
W	W	-	Y	-	Y
-	Z	-	-	Z	Z

A box defined as an array of 5x6

BOX-2

Z	W	-
W	Y	-
Y	Z	Y
-	Z	-
-	-	-

A box defined as an array of 5x3

Resulting mem

0	3	7	8
2	5	9	10
7	8	11	12
10	10	10	11
14	14	13	14
15	15	14	15
18	18	17	18

(e)

BOX-1

A	B	C	A	B	C
B	C	C	A	-	A

A box defined as an array of 2x6

BOX-2

A	B	B	A
B	A	-	C

A box defined as an array of 2x4

Resulting mem

0	2	4	5	7
2	0	2	3	5
4	2	1	2	4
6	4	3	4	3
8	6	5	6	5
9	7	6	5	6
11	9	8	7	7

(f)

BOX-1

-	M	V	R
M	-	R	-
R	M	M	-
V	M	-	R

A box defined as an array of 4x4

BOX-2

M	-	-	V
V	R	R	M
-	-	-	-
-	M	-	R

A box defined as an array of 4x4

Resulting mem

0	2	4	5	8
3	5	6	7	8
6	8	9	10	11
9	10	12	13	13
11	12	14	15	15

Your task is to convert the first box into the second box by using some operations resulting in the minimum cost. The conversion rules and operations are defined as follows:

- You should compare the boxes column by column. Each conversion operation is column-wise.
- A column could be deleted completely. The **deletion operation** costs as much as the number of full cells in the column. For instance; if the column consists of 5 cells where 3 of them full and 2 of them are empty, then deleting that column costs 3 units.
- For a column of Box-2, a new corresponding column could be inserted into Box-1 at any location (between two columns or as the initial column or as the final column). The **insertion operation** costs as much as the number of full cells inside the new column. For instance; if the newly inserted column consists of 5 cells where 3 of them full and 2 of them are empty, then inserting that column costs 3 units.
- A column could be converted into a new column by reordering its cells. For intance, if a column consists of 5 cells including ['X', 'A', '-', 'B', '-'], it can be reordered as ['A', '-', '-', 'B', 'X']. The **reordering operation** costs as much as the number of cells whose locations are changed. For the example given, since the locations of the cells including 'A', 'X' and '-' changed only, it costs 3 units.
- A column could be converted into a new column by replacing its cells with some other cells. For the **replacement operation**, if a full cell is replaced with some other full cell, then it costs 1 unit. However, if an empty cell is replaced with a full cell, or vice versa, then it costs 2 units. For instance, if a column consists of 5 cells including ['X', 'A', '-', 'B', '-'], its cells can be replaced as ['X', 'C', 'D', '-', '-'], it costs <change from 'A' to 'C'> + <change from '-' to 'D'> + <change from 'B' to '-'> = 1 + 2 + 2 = 5 units.

- Each operation is independent from each other. At each transition, apply only one of them.
- **HINT:** You should implement the dynamic programming column-wise. That is, for each column of Box-2, consider a corresponding column inside Box-1 which has been obtained by the operations above. The way of how to apply memoization is explained in the following parts.

Example IO:

1) Given boxes in (a) of the above Figure:

- return value (i.e. min cost) is 8.
- Since this is the first example, let's explain all the cells of mem array:

mem[0][0] : *Conversion of no columns of box1 to no columns of box2*
No operation
=> costs 0

mem[0][1] : *Conversion of no columns of box1 to first column of box2*
Apply insertion operation to obtain the first column of box2
=> costs 3

mem[0][2] : *Conversion of no columns of box1 to first 2 columns of box2*
Apply insertion operation for both of the first two columns of box2
=> costs 3 +2 = 5

mem[0][3] : *Conversion of no columns of box1 to first 3 columns of box2*
Apply insertion operation for each of the 3 columns of box2
=> costs 3 + 2 + 3 = 8

mem[1][0] : *Conversion of first column of box1 to no columns of box2*
Apply deletion operation on the initial column of box1
=> costs 3

mem[1][1] : *Conversion of first column of box1 to first column of box2*
Apply reordering operation to change the first column of box1 to the first column of box2
=> costs 4

mem[1][2] : *Conversion of first column of box1 to first 2 columns of box2*
Apply reordering operation to change the first column of box1 to the first column of box2 and
Apply insertion operation to obtain the second column of box2
=> costs 4 + 2 = 6

mem[1][3] : *Conversion of first column of box1 to first 3 columns of box2*
Apply insertion operation to obtain the first column of box2 and
Apply insertion operation to obtain the second column of box2 and
Apply reordering operation to change the first column of box1 to the third column of box2
=> costs 3 +2 + 3 = 8

2) Given boxes in (b) of the above Figure:

- return value (i.e. min cost) is 6.
- at dynamic programming, final mem array is given its right side.

3) Given boxes in (c) of the above Figure:

- return value (i.e. min cost) is 19.
- at dynamic programming, final mem array is given its right side.

4) Given boxes in (d) of the above Figure:

- return value (i.e. min cost) is 18.

- at dynamic programming, final mem array is given its right side.

5) Given boxes in (e) of the above Figure:

- return value (i.e. min cost) is 7.
- at dynamic programming, final mem array is given its right side.

6) Given boxes in (f) of the above Figure:

- return value (i.e. min cost) is 15.
- at dynamic programming, final mem array is given its right side.

Implementation:

You will implement only one function for solution of that problem:

- Dynamic programming in ***dp_sln()***

The function is expected to **return** the answer to the given problem which is **the minimum cost of operations.** Return **only** the min cost value and nothing more.

The ***char**& box1*** and ***char**& box2*** variables are the parameters which pass the input 2D array of boxes to your functions. **Do not modify those arrays!** The format of boxes will be as stated in the problem definition above.

The ***int nrow***, ***int ncol1*** and ***int ncol2*** variables are the parameters which passes the number of rows of both boxes, number of columns of *box1* and number of columns of *box2,* repectively, to your function.

You should use ***int**& mem*** variable (i.e. array), which is the last parameter at definition of the function, as **the array of memoized values**. For *dp_sln()* function, final values in the *mem* variable will be considered for grading. Note that it is a 2D array. It is defined as **the size of (ncol1+1) x (ncol2+1**) such that its rows correspond to columns of *box1* and its columns correspond to columns of *box2*. **That is, the *mem[i][j]* will be used to indicate the TOTAL COST of matching of THE FIRST i columns of box1 with THE FIRST j columns of box2**. Thus *mem[0][0]* indicates there is no matching columns in box1 and box2! While testing and grading, all the cells of *mem* array will be initialized to -1's. So, while implementing your function, **you can assume that *mem* is an array of array of -1's. Do not return that variable/array.**

The *dp_sln()* function should be implemented with bottom-up (iterative) approach.

Implement the function in most efficient way.