[All Tracks](https://www.hackerearth.com/practice/)**-->**[Data Structures](https://www.hackerearth.com/practice/data-structures/) **-->**[Arrays](https://www.hackerearth.com/practice/data-structures/arrays/)**-->** [Multi-dimensional](https://www.hackerearth.com/practice/data-structures/arrays/multi-dimensional/)Problem **-->**

**The Wealthy Landlord**

Tag(s):Easy,Math

1. Easy, Math

Shyam Lal, a wealthy landlord from the state of Rajasthan, being an old fellow and tired of doing hard work, decided to sell all his farmland and to live rest of his life with that money. No other farmer is rich enough to buy all his land so he decided to partition the land into rectangular plots of different sizes with different cost per unit area. So, he sold these plots to the farmers but made a mistake. Being illiterate, he made partitions that could be overlapping. When the farmers came to know about it, they ran to him for compensation of extra money they paid to him. So, he decided to return all the money to the farmers of that land which was overlapping with other farmer's land to settle down the conflict. All the portion of conflicted land will be taken back by the landlord.  
  
To decide the total compensation, he has to calculate the total amount of money to return back to farmers with the same cost they had purchased from him. Suppose, Shyam Lal has a total land area of **1000 x 1000** equal square blocks where each block is equivalent to a unit square area which can be represented on the co-ordinate axis. Now find the total amount of money, he has to return to the farmers. Help Shyam Lal to accomplish this task.

**Input Format:**  
The first line of the input contains an integer **N**, denoting the total number of land pieces he had distributed. Next **N** line contains the **5** space separated integers (**X1**, **Y1**), (**X2**, **Y2**) to represent a rectangular piece of land, and cost per unit area **C**.  
(**X1**, **Y1**) and (**X2**, **Y2**) are the locations of first and last square block on the diagonal of the rectangular region.

**Output Format:**  
Print the total amount he has to return to farmers to solve the conflict.

**Constraints:**  
1 ≤ **N** ≤ 100  
1 ≤ **X1** ≤ **X2** ≤ 1000  
1 ≤ **Y1** ≤ **Y2** ≤ 1000  
1 ≤ **C** ≤ 1000

**SAMPLE INPUT**

3

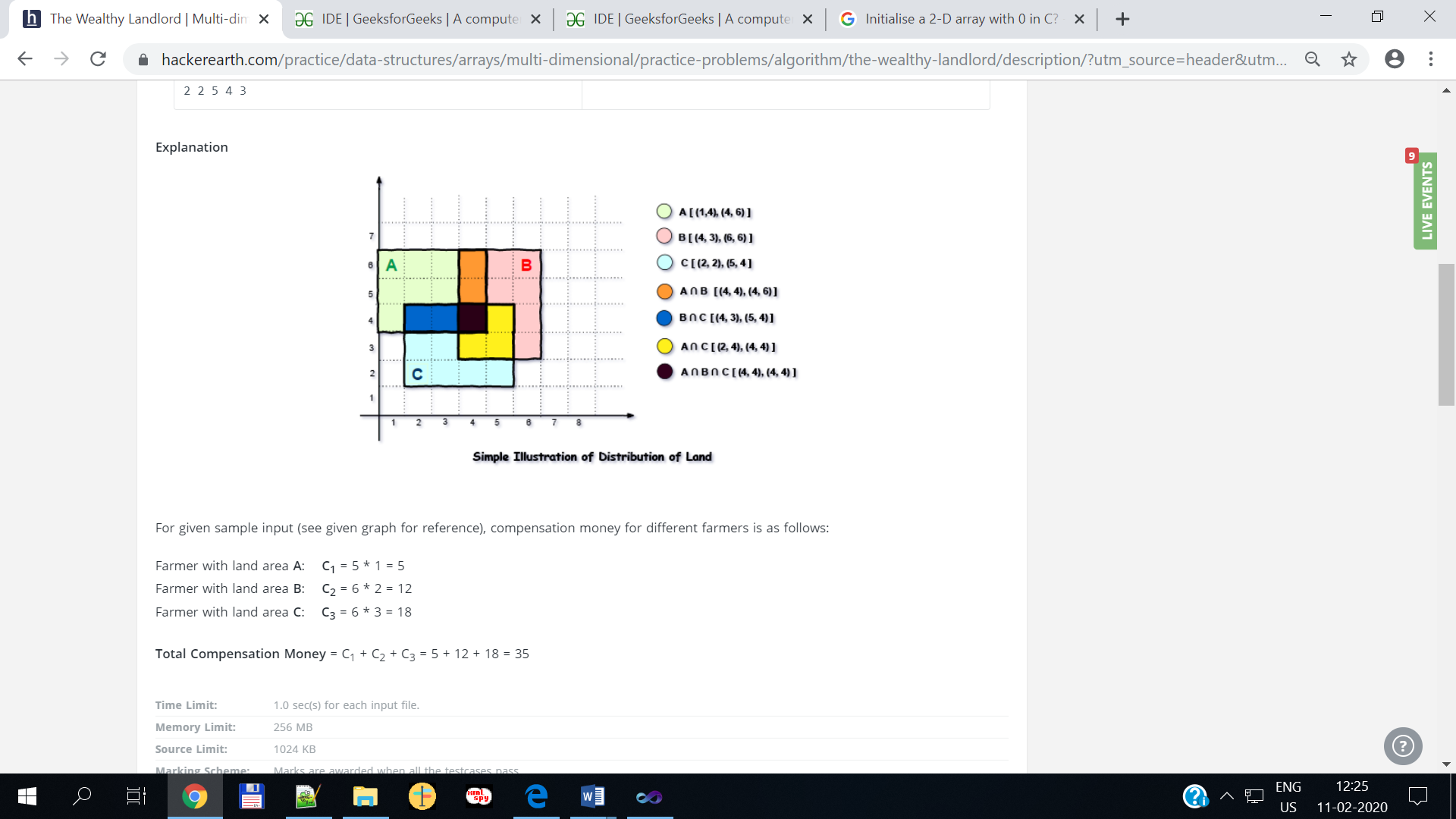
1 4 4 6 1

4 3 6 6 2

2 2 5 4 3

**SAMPLE OUTPUT**

35



//My C Solution :

#include <stdio.h>

#define LEN 1001

#define MAX(a,b) (a>b)?(a):(b)

int maxX =0;

int maxY =0;

long int Land[LEN][LEN]= { 0 } ;

void compensate(){

int i,j;

long sum=0;

for(i=0;i<maxX;i++)

for(j=0;j<maxY;j++)

if(Land[i][j]<0)

sum+=Land[i][j];

printf("%ld\n",((-1)\*sum));

}

int main(){

int i,j,cost,N,x1,y1,x2,y2;

scanf("%d",&N);

while(N--){

scanf("%d%d%d%d%d",&x1,&y1,&x2,&y2,&cost);

if((x1>maxX)||(x2>maxX))

maxX=MAX(x1,x2);

if((y1>maxY)||(y2>maxY))

maxY=MAX(y1,y2);

for(i=x1-1; i<x2; i++) {

for(j=y1-1; j<y2; j++) {

if(Land[i][j]==0)

Land[i][j]=cost;

else {

if(Land[i][j]>0)

Land[i][j]=(-1)\*(Land[i][j]+cost);

else

Land[i][j]=Land[i][j]-cost;

}

}

}

}

compensate();

return 0;

}

/\*Best Submissions :

C :

1. *#include* <stdio.h>
2. *#include* <stdlib.h>
3. *#define* SIZE 1001
5. int main(void){
6. int i, j;
7. int n, x1, x2, y1, y2, cost = 0;
8. long long total\_cost = 0;
9. int arr[SIZE][SIZE] = {0};
10. *//let the 2D Array represents the field of Shyam Lal*
11. scanf("%d", &n);
13. while(n--){
15. scanf("%d%d%d%d", &x1, &y1, &x2, &y2);
16. scanf("%d", &cost);
18. for(i = x1; i <= x2; i++){
19. for(j = y1; j <= y2; j++){
20. if(arr[i][j] == 0)
21. arr[i][j] += cost;
23. else if(arr[i][j] > 0)
24. arr[i][j] = (-1)\*(arr[i][j] + cost);
26. else if(arr[i][j] < 0)
27. arr[i][j] = arr[i][j] - cost;
28. }
29. }
30. }
32. for(i = 1; i < SIZE; i++){
33. for(j = 1; j < SIZE; j++){
34. if(arr[i][j] < 0)
35. total\_cost += arr[i][j];
36. }
37. }
38. *//Print the absolute value of result(total\_cost)*
39. printf("%lld\n", (-1) \* total\_cost);
41. return 0;
42. }

C++ :

1. *#include* <iostream>
2. using namespace std;
4. int main()
5. {
6. int n;
7. int x1,x2,y1,y2;
8. int p;
9. int a[1000][1000]={0};
10. cin>>n;
11. long sum=0;
12. while(n--)
13. {
14. cin>>x1>>y1>>x2>>y2>>p;
15. for(int i=x1-1;i<x2;i++)
16. {
17. for(int j=y1-1;j<y2;j++)
18. {
19. if(a[i][j]==0)
20. a[i][j]=p;
21. else if(a[i][j]>0)
22. a[i][j]=(-1)\*(a[i][j]+p);
23. else
24. a[i][j]=a[i][j]-p;
26. }
27. }
29. }
30. for(int i=0;i<1000;i++)
31. for(int j=0;j<1000;j++)
32. if(a[i][j]<0)
33. sum+=a[i][j];
35. cout<<-1\*sum<<endl;
36. return 0;
37. }

C++ 14 :

1. *#include* <bits/stdc++.h>
2. using namespace std;
4. struct Element {
5. bool doubly = false;
6. long long cost = 0;
7. };
9. int main()
10. {
11. int N;
12. int X1, Y1, X2, Y2, C;
14. cin >> N;
15. Element matrix[1001][1001];
17. for(int i = 0; i < N; i++)
18. {
19. cin >> X1 >> Y1 >> X2 >> Y2 >> C;
21. for(int i = Y1; i <= Y2; i++) {
22. for(int j = X1; j <= X2; j++) {
23. if(matrix[i][j].cost > 0) {
24. matrix[i][j].doubly = true;
25. }
26. matrix[i][j].cost += C;
27. }
28. }
29. }
31. long long totalSum = 0;
33. for(int i = 1; i <= 1000; i++) {
34. for(int j = 1; j <= 1000; j++) {
35. if(matrix[i][j].doubly) {
36. totalSum += matrix[i][j].cost;
37. }
38. }
39. }
41. cout << totalSum << endl;
43. return 0;
44. }

C# :

1. using System;
2. using System.Collections;
3. using System.Collections.Generic;
4. using System.Linq;
5. using System.Numerics;
7. class MyClass
8. {
9. struct Rect
10. {
11. public int x1;
12. public int y1;
13. public int x2;
14. public int y2;
15. public int compen;
17. }
18. static void Main()
19. {
20. var n = Int32.Parse(Console.ReadLine());
21. int[,] land = new int[1001,1001];
22. Rect[] rects = new Rect[n];
23. foreach(var i in Enumerable.Range(0, n))
24. {
25. var line = Console.ReadLine().Split(' ').Select(a => Int32.Parse(a));
26. var x1 = line.ElementAt(0);
27. var y1 = line.ElementAt(1);
28. var x2 = line.ElementAt(2);
29. var y2 = line.ElementAt(3);
30. rects[i] = new Rect
31. {
32. x1 = x1,
33. y1 = y1,
34. x2 = x2,
35. y2 = y2,
36. compen = line.ElementAt(4)
37. };
38. for(int x = x1; x <= x2; x++)
39. {
40. for (int y = y1; y <= y2; y++)
41. {
42. land[x, y]++;
43. }
44. }
45. }
46. var com = 0L;
47. foreach (var i in Enumerable.Range(0, n))
48. {
49. for (int x = rects[i].x1; x <= rects[i].x2; x++)
50. {
51. for (int y = rects[i].y1; y <= rects[i].y2; y++)
52. {
53. if(land[x, y] != 1)
54. {
55. com += rects[i].compen;
56. }
57. }
58. }
59. }
60. Console.Write(com);
61. }
63. }

Java :

1. import java.io.\*;
2. import java.util.\*;
4. class InputReader {
5. private InputStream stream;
6. private byte[] buf = new byte[1024];
7. private int curChar;
8. private int numChars;
9. private SpaceCharFilter filter;
11. public InputReader(InputStream stream) {
12. this.stream = stream;
13. }
15. public int read() {
16. if (numChars == -1)
17. throw new InputMismatchException();
18. if (curChar >= numChars) {
19. curChar = 0;
20. try {
21. numChars = stream.read(buf);
22. } catch (IOException e) {
23. throw new InputMismatchException();
24. }
25. if (numChars <= 0)
26. return -1;
27. }
28. return buf[curChar++];
29. }
31. public int readInt() {
32. int c = read();
33. while (isSpaceChar(c))
34. c = read();
35. int sgn = 1;
36. if (c == '-') {
37. sgn = -1;
38. c = read();
39. }
40. int res = 0;
41. do {
42. if (c < '0' || c > '9')
43. throw new InputMismatchException();
44. res \*= 10;
45. res += c - '0';
46. c = read();
47. } while (!isSpaceChar(c));
48. return res \* sgn;
49. }
51. public String readString() {
52. int c = read();
53. while (isSpaceChar(c))
54. c = read();
55. StringBuilder res = new StringBuilder();
56. do {
57. res.appendCodePoint(c);
58. c = read();
59. } while (!isSpaceChar(c));
60. return res.toString();
61. }
63. public boolean isSpaceChar(int c) {
64. if (filter != null)
65. return filter.isSpaceChar(c);
66. return c == ' ' || c == '\n' || c == '\r' || c == '\t' || c == -1;
67. }
69. public String next() {
70. return readString();
71. }
73. public interface SpaceCharFilter {
74. public boolean isSpaceChar(int ch);
75. }
76. }
78. class OutputWriter {
79. private final PrintWriter writer;
81. public OutputWriter(OutputStream outputStream) {
82. writer = new PrintWriter(new BufferedWriter(new OutputStreamWriter(outputStream)));
83. }
85. public OutputWriter(Writer writer) {
86. this.writer = new PrintWriter(writer);
87. }
89. public void print(Object...objects) {
90. for (int i = 0; i < objects.length; i++) {
91. if (i != 0)
92. writer.print(' ');
93. writer.print(objects[i]);
94. }
95. }
97. public void printLine(Object...objects) {
98. print(objects);
99. writer.println();
100. }
102. public void close() {
103. writer.close();
104. }
106. public void flush() {
107. writer.flush();
108. }
110. }
112. class IOUtils {
113. public static int[] readIntArray(InputReader in, int size) {
114. int[] array = new int[size];
115. for (int i = 0; i < size; i++)
116. array[i] = in.readInt();
117. return array;
118. }
119. }
121. class TestClass
122. {
123. public static void main(String args[] ) throws Exception
124. {
125. InputReader in = new InputReader(System.in);
126. OutputWriter out = new OutputWriter(System.out);
128. *// int n = in.readInt();*
129. *// String s = in.readString();*
130. *// int[] x = IOUtils.readIntArray(in,N);*
131. *// out.printLine("X");*

134. *//PROGRAM*
135. int t = in.readInt();
136. int cost[][] = new int[1005][1005];
137. int freq[][] = new int[1005][1005];
138. int x1,y1,x2,y2,c,i,j;
139. while(t-->0)
140. {
141. x1 = in.readInt();
142. y1 = in.readInt();
143. x2 = in.readInt();
144. y2 = in.readInt();
145. c = in.readInt();
147. for(i=x1;i<=x2;i++)
148. {
149. for(j=y1;j<=y2;j++)
150. {
151. freq[i][j]++;
152. cost[i][j] += c;
153. }
154. }
155. }
156. long sum=0;
157. for(i=0;i<1005;i++)
158. {
159. for(j=0;j<1005;j++)
160. {
161. if(freq[i][j]>1)
162. sum+=cost[i][j];
163. }
164. }
165. out.printLine(sum);
166. out.flush();
167. out.close();
168. }
169. }

Java 8 :

1. import java.io.DataInputStream;
2. import java.io.FileInputStream;
3. import java.io.IOException;
5. class TestClass {
6. public static void main(String[] args) throws IOException {
7. Reader s = new Reader();
8. int T = s.nextInt();
9. long money=0;
10. int[][] land = new int[1001][1001];
11. while (T-- > 0) {
12. int x1 = s.nextInt();
13. int y1 = s.nextInt();
14. int x2 = s.nextInt();
15. int y2 = s.nextInt();
16. int cost = s.nextInt();
18. for(int i=x1;i<=x2;i++){
19. for(int j=y1;j<=y2;j++){
20. if(land[i][j]==-1){
21. money += cost;
22. }else if(land[i][j]==0){
23. land[i][j] = cost;
24. }else{
25. money += land[i][j] + cost;
26. land[i][j] = -1;
27. }
28. }
29. }
30. }
31. System.out.println(money);
32. }
34. static class Reader {
35. final private int BUFFER\_SIZE = 1 << 16;
36. private DataInputStream din;
37. private byte[] buffer;
38. private int bufferPointer, bytesRead;
40. public Reader() {
41. din = new DataInputStream(System.in);
42. buffer = new byte[BUFFER\_SIZE];
43. bufferPointer = bytesRead = 0;
44. }
46. public Reader(String file\_name) throws IOException {
47. din = new DataInputStream(new FileInputStream(file\_name));
48. buffer = new byte[BUFFER\_SIZE];
49. bufferPointer = bytesRead = 0;
50. }
52. public String readLine() throws IOException {
53. byte[] buf = new byte[64]; *// line length*
54. int cnt = 0, c;
55. while ((c = read()) != -1) {
56. if (c == '\n')
57. break;
58. buf[cnt++] = (byte) c;
59. }
60. return new String(buf, 0, cnt);
61. }
63. public int nextInt() throws IOException {
64. int ret = 0;
65. byte c = read();
66. while (c <= ' ')
67. c = read();
68. boolean neg = (c == '-');
69. if (neg)
70. c = read();
71. do {
72. ret = ret \* 10 + c - '0';
73. } while ((c = read()) >= '0' && c <= '9');
74. if (neg)
75. return -ret;
76. return ret;
77. }
79. public long nextLong() throws IOException {
80. long ret = 0;
81. byte c = read();
82. while (c <= ' ')
83. c = read();
84. boolean neg = (c == '-');
85. if (neg)
86. c = read();
87. do {
88. ret = ret \* 10 + c - '0';
89. } while ((c = read()) >= '0' && c <= '9');
90. if (neg)
91. return -ret;
92. return ret;
93. }
95. public double nextDouble() throws IOException {
96. double ret = 0, div = 1;
97. byte c = read();
98. while (c <= ' ')
99. c = read();
100. boolean neg = (c == '-');
101. if (neg)
102. c = read();
103. do {
104. ret = ret \* 10 + c - '0';
105. } while ((c = read()) >= '0' && c <= '9');
106. if (c == '.')
107. while ((c = read()) >= '0' && c <= '9')
108. ret += (c - '0') / (div \*= 10);
109. if (neg)
110. return -ret;
111. return ret;
112. }
114. private void fillBuffer() throws IOException {
115. bytesRead = din.read(buffer, bufferPointer = 0, BUFFER\_SIZE);
116. if (bytesRead == -1)
117. buffer[0] = -1;
118. }
120. private byte read() throws IOException {
121. if (bufferPointer == bytesRead)
122. fillBuffer();
123. return buffer[bufferPointer++];
124. }
126. public void close() throws IOException {
127. if (din == null)
128. return;
129. din.close();
130. }
131. }
132. }

PHP :

1. <?php
3. fscanf(STDIN, "%d\n", $tc);
4. $cost =array();
5. $farm = array();
6. $tm = 0;
7. for($i=0;$i<$tc;$i++){
8. $t = explode(" ",fgets(STDIN));
9. $cost[$i] = $t[4];
10. for($j=$t[0];$j<=$t[2];$j++){
11. for($k=$t[1];$k<=$t[3];$k++){
12. if(!isset($farm[$j][$k])){
13. $farm[$j][$k] = a."$i";
14. }else{
15. if($farm[$j][$k] == "f"){
16. $tm = $tm + $cost[$i];
17. }else{
18. $tm = $tm + $cost[$i];
19. $c = substr($farm[$j][$k],1);
20. $tm = $tm + $cost[$c];
21. $farm[$j][$k] = "f";
22. }
23. }
24. }
25. }
26. }
28. echo $tm;
30. ?>

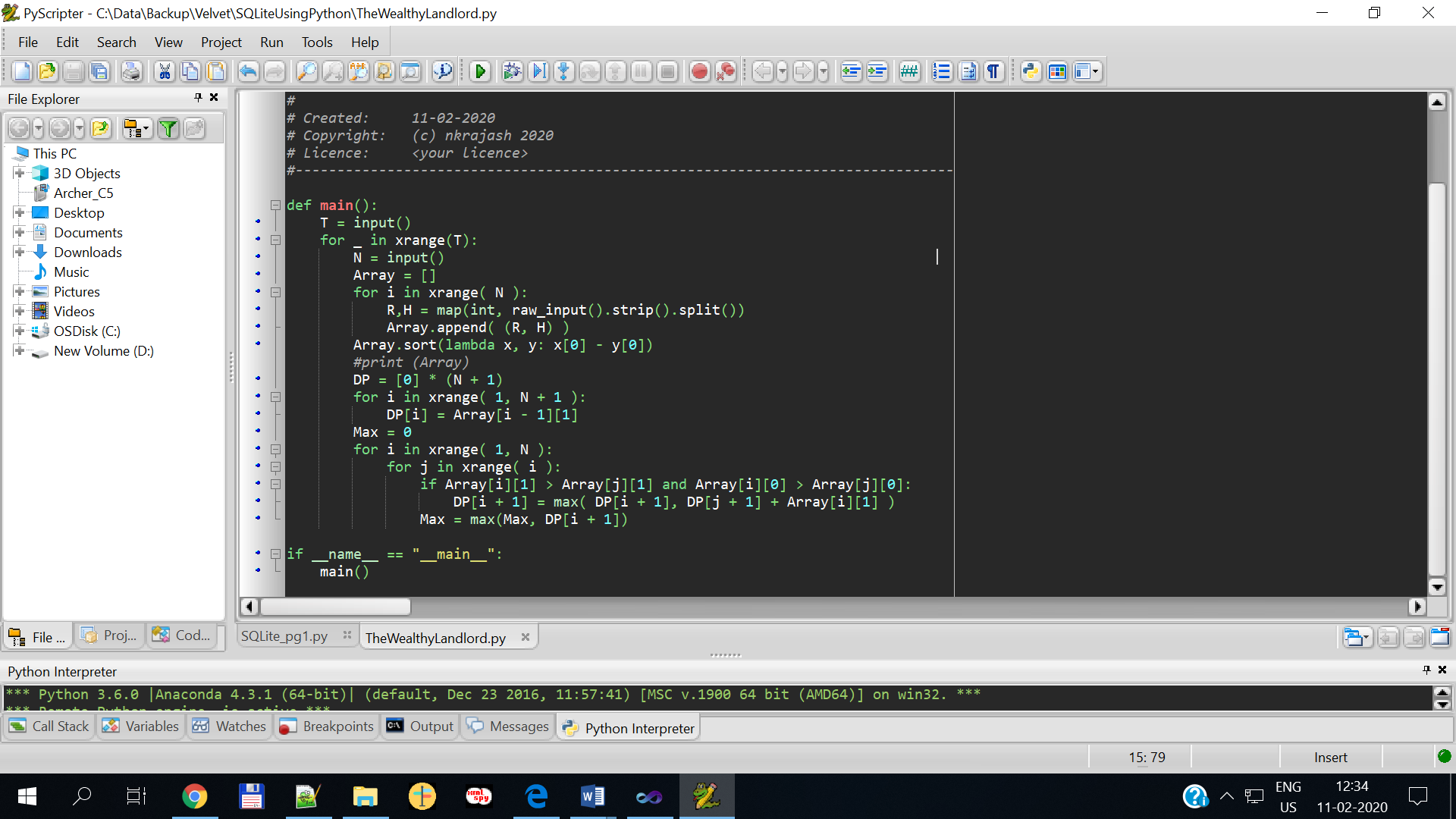
Python :

1. def solution3():
2. try:
3. N = int(raw\_input())
4. except:
5. return 0
7. SZ = 1001
8. grid = [[0] \* SZ for i in xrange(SZ)]
9. min\_x = min\_y = float("inf")
10. max\_x = max\_y = float("-inf")
11. for i in xrange(N):
12. x1, y1, x2, y2, c = map(int, raw\_input().split())
13. min\_x = min(min\_x, x1)
14. min\_y = min(min\_y, y1)
15. max\_x = max(max\_x, x2)
16. max\_y = max(max\_y, y2)
17. for i in xrange(x1, x2 + 1):
18. for j in xrange(y1, y2 + 1):
19. if grid[i][j] == 0:
20. grid[i][j] = c
21. elif grid[i][j] > 0:
22. grid[i][j] = -(grid[i][j] + c)
23. else:
24. grid[i][j] -= c
26. cost = 0
27. for i in xrange(min\_x, max\_x + 1):
28. for j in xrange(min\_y, max\_y + 1):
29. if grid[i][j] < 0:
30. cost += grid[i][j]
32. return -cost
34. print solution3()

Python 3 :

1. def solution3():
2. try:
3. N = int(input())
4. except:
5. return 0
7. SZ = 1001
8. grid = [[0] \* SZ for i in range(SZ)]
9. min\_x = min\_y = float("inf")
10. max\_x = max\_y = float("-inf")
11. for i in range(N):
12. x1, y1, x2, y2, c = map(int, input().split())
13. min\_x = min(min\_x, x1)
14. min\_y = min(min\_y, y1)
15. max\_x = max(max\_x, x2)
16. max\_y = max(max\_y, y2)
17. for i in range(x1, x2 + 1):
18. for j in range(y1, y2 + 1):
19. if grid[i][j] == 0:
20. grid[i][j] = c
21. elif grid[i][j] > 0:
22. grid[i][j] = -(grid[i][j] + c)
23. else:
24. grid[i][j] -= c
26. cost = 0
27. for i in range(min\_x, max\_x + 1):
28. for j in range(min\_y, max\_y + 1):
29. if grid[i][j] < 0:
30. cost += grid[i][j]
32. return -cost
34. print( solution3())

Alt Solution :



Ruby :

1. s=0
2. t=[]
3. 1000.times{
4. t<<[0]\*1000
5. }
6. gets.to\_i.times{
7. a,b,c,d,e=gets.split.map &:to\_i
8. (a-1).upto(c-1){|i|
9. (b-1).upto(d-1){|j|
10. if t[j][i]==0
11. t[j][i]=e
12. elsif t[j][i]>0
13. s+=t[j][i]
14. t[j][i]=-1
15. s+=e
16. elsif t[j][i]<0
17. s+=e
18. end
19. }
20. }
21. }
22. p s

\*/

/\*Editorial :

#include <stdio.h>

#include <stdlib.h>

#define SIZE 1001

int main(void){

int i, j;

int n, x1, x2, y1, y2, cost = 0;

long long total\_cost = 0;

int arr[SIZE][SIZE] = {0};

//let the 2D Array represents the field of Shyam Lal

scanf("%d", &n);

while(n--){

scanf("%d%d%d%d", &x1, &y1, &x2, &y2);

scanf("%d", &cost);

for(i = x1; i <= x2; i++){

for(j = y1; j <= y2; j++){

if(arr[i][j] == 0)

arr[i][j] += cost;

else if(arr[i][j] > 0)

arr[i][j] = (-1)\*(arr[i][j] + cost);

else if(arr[i][j] < 0)

arr[i][j] = arr[i][j] - cost;

}

}

}

for(i = 1; i < SIZE; i++){

for(j = 1; j < SIZE; j++){

if(arr[i][j] < 0)

total\_cost += arr[i][j];

}

}

//Print the absolute value of result(total\_cost)

printf("%lld\n", (-1) \* total\_cost);

return 0;

}

\*/

/\*I/O :

In #1 :

1

48 12 49 27 8

Out #1 :

0

In #2 :

3

88 34 99 76 44

82 65 94 100 81

58 16 65 66 7

Out #2 :

10500

In #3 :

63

3 84 69 93 37

28 59 42 93 17

31 33 67 45 83

64 23 71 76 62

18 15 92 92 67

32 35 74 90 47

78 59 99 76 66

19 32 94 89 53

77 49 83 52 63

79 15 85 22 91

71 84 78 85 11

95 64 97 65 75

87 57 92 67 65

20 91 76 98 61

60 57 87 69 98

94 17 97 61 37

43 54 85 64 24

62 67 88 76 97

9 62 84 87 41

6 38 13 78 26

98 42 99 82 74

11 91 31 93 65

77 62 81 68 97

20 17 26 83 4

43 20 73 85 32

32 89 86 98 21

15 86 41 98 36

93 85 95 96 16

55 15 62 88 16

3 15 24 79 85

92 61 98 100 54

55 47 75 90 82

95 58 100 65 3

66 10 90 55 22

52 28 66 42 62

52 17 70 28 10

45 57 63 92 23

37 21 47 92 14

80 68 95 77 95

12 62 76 74 87

58 48 65 71 11

92 34 95 80 65

66 47 87 54 24

1 1 79 68 80

78 93 92 94 41

35 90 37 94 33

79 40 91 63 93

53 97 61 99 9

15 53 30 58 68

71 15 74 29 26

67 75 100 90 95

42 19 62 65 66

17 87 84 96 84

21 16 73 24 16

5 63 12 77 23

68 70 83 74 66

96 45 98 89 21

38 31 93 63 78

39 20 84 47 46

70 85 71 91 14

12 46 65 89 49

69 71 98 89 54

11 39 95 97 36

Out #3 :

2659214

In #4 :

100

265 84 385 531 154

467 71 634 683 884

3 853 947 934 658

856 698 933 762 374

53 427 524 670 884

793 151 840 584 241

571 828 712 853 910

637 623 860 977 81

189 796 755 838 582

105 843 451 947 52

239 77 623 358 633

114 721 626 834 785

178 34 995 988 665

155 43 508 263 63

994 186 995 842 314

812 326 953 900 131

95 110 787 673 41

636 444 958 547 974

659 497 969 672 993

140 546 995 981 112

467 498 496 556 802

618 625 869 892 746

399 703 690 745 362

875 930 892 1000 31

323 538 484 868 484

784 383 969 903 655

444 373 921 881 716

711 699 960 720 623

400 400 697 896 799

556 116 649 417 770

969 199 992 810 923

936 814 991 890 788

239 416 572 421 153

122 723 954 953 470

675 1 802 675 59

680 688 875 913 876

761 320 879 719 415

583 889 853 905 825

34 672 510 762 640

817 399 988 738 798

282 602 641 914 621

570 496 805 988 596

253 712 385 779 549

558 39 566 41 26

58 194 881 212 78

607 408 749 588 205

835 729 870 927 138

258 674 922 931 423

723 610 735 621 374

574 717 1000 870 751

873 145 896 755 725

341 585 879 601 92

683 856 897 893 327

707 426 758 928 300

401 409 910 422 504

335 878 337 977 786

543 886 616 913 422

352 331 420 484 70

119 66 369 185 43

499 906 953 986 307

774 270 915 359 149

846 335 886 972 256

305 857 545 910 733

624 871 957 929 28

158 749 459 890 519

453 864 730 900 655

203 816 725 847 547

373 145 468 995 658

33 434 759 486 144

438 652 535 658 371

667 221 880 660 161

94 34 457 98 516

980 524 985 830 201

937 219 977 920 462

819 761 838 930 139

170 908 589 927 180

168 247 377 283 70

846 933 893 946 306

813 684 850 713 473

925 883 980 953 782

567 234 881 984 395

213 443 771 655 940

888 964 933 981 478

465 431 675 952 597

906 227 935 970 399

289 988 541 993 970

182 13 388 891 311

417 642 604 945 5

492 806 544 913 130

139 596 637 897 304

863 721 925 768 55

966 792 973 930 254

924 80 972 302 84

109 496 167 591 387

126 420 704 967 358

636 51 685 371 243

659 859 870 949 459

165 427 496 704 755

35 442 225 698 55

744 389 762 869 261

Out #4 :

2823896365

In #5 :

35

17 23 82 46 38

91 17 99 66 81

3 7 42 20 91

1 97 49 98 37

44 11 45 87 95

97 11 99 14 72

42 2 54 89 15

23 68 24 99 3

48 50 55 97 54

73 39 82 58 72

7 86 73 96 18

67 31 87 40 83

3 80 34 83 8

97 55 100 57 15

92 49 96 86 28

92 95 93 99 62

73 68 100 98 45

88 73 92 93 43

49 65 50 82 34

6 4 11 87 88

19 30 27 62 8

78 91 95 98 1

92 78 100 80 9

99 18 100 82 9

66 68 95 94 32

19 51 81 77 56

7 11 97 96 19

63 44 68 99 53

42 53 50 80 20

63 18 99 25 56

27 30 39 78 58

34 90 35 92 21

9 47 92 87 63

40 93 56 96 46

12 49 61 53 63

Out #5 :

884348

In #6:

70

302 238 461 382 193

107 75 148 469 261

120 21 388 200 285

200 447 340 476 441

250 338 330 427 217

51 32 308 200 339

165 272 472 369 134

157 36 240 369 407

36 336 413 418 182

491 486 495 497 321

292 405 490 489 57

351 167 486 472 368

245 253 414 490 59

1 324 155 353 326

156 234 228 236 31

240 81 491 462 435

277 32 401 488 285

85 448 91 490 467

21 200 342 353 407

150 73 290 308 447

356 472 482 483 184

8 275 349 318 130

79 224 428 498 96

235 390 279 491 410

321 441 352 475 181

269 12 421 378 61

84 395 486 422 493

352 460 454 495 33

403 87 446 425 167

242 273 438 355 105

350 384 448 457 470

356 275 420 398 334

171 379 284 429 217

458 479 479 498 142

284 7 364 288 17

89 369 138 445 40

361 186 420 488 483

395 200 449 231 315

130 482 256 494 500

371 346 498 430 478

252 236 359 438 84

431 122 472 331 156

19 248 453 383 231

63 449 236 497 491

132 423 297 455 310

60 385 189 492 314

473 460 486 469 379

116 423 394 491 300

448 299 455 316 313

497 99 499 456 160

111 209 247 277 309

486 339 490 372 83

3 303 344 454 457

22 101 147 138 200

2 197 273 368 438

64 259 178 293 55

368 27 499 298 159

5 456 408 477 220

287 336 435 438 404

74 12 494 86 111

205 191 492 428 297

130 371 307 479 118

51 404 442 408 207

118 445 367 467 199

85 145 376 269 95

292 233 299 370 439

192 338 202 389 66

354 45 464 478 398

173 184 441 303 392

292 144 324 256 99

Out #6 :

379863074

In #7 :

100

367 330 383 498 188

12 50 77 684 750

717 350 727 579 16

491 792 732 800 411

152 229 702 613 275

549 267 575 746 548

53 49 727 417 435

278 276 545 569 791

57 629 795 754 45

339 239 688 721 732

418 5 572 299 497

8 687 702 702 278

474 794 501 798 694

548 345 765 425 103

554 108 659 249 536

153 215 770 361 120

355 44 625 674 426

552 164 590 542 119

527 482 696 585 610

467 325 713 460 766

768 489 775 630 721

195 754 433 764 624

573 72 661 483 111

638 717 743 776 632

430 79 573 240 62

79 368 204 613 431

292 574 719 757 497

718 94 756 325 729

620 636 705 775 634

656 253 671 668 799

723 304 779 567 455

476 430 494 600 798

202 415 243 607 203

728 390 749 718 345

364 97 438 519 14

109 110 630 288 387

319 753 563 772 582

111 210 732 392 113

56 658 548 741 248

285 351 631 564 149

182 627 251 769 360

116 734 351 796 381

651 64 789 275 77

155 509 561 575 741

671 226 706 757 147

435 743 580 752 155

369 429 769 521 418

80 134 371 430 105

83 740 611 772 623

176 534 269 749 649

143 287 722 296 308

694 274 753 710 161

522 662 745 721 544

186 333 650 514 193

563 301 656 543 539

602 745 793 775 781

360 704 512 723 125

267 639 269 726 72

705 642 781 733 778

70 489 410 492 663

352 471 359 747 770

512 518 679 769 207

36 169 479 702 156

289 610 585 691 22

407 12 449 498 646

399 217 466 313 747

610 134 779 534 200

467 126 724 800 321

95 215 164 358 472

99 348 192 661 379

423 53 462 677 344

379 286 482 399 411

392 325 702 499 344

629 562 690 706 422

432 685 446 773 183

419 300 533 396 674

241 604 485 612 486

553 28 569 121 313

637 145 785 414 685

6 34 542 166 380

389 142 517 689 583

257 743 530 748 643

24 123 364 160 66

46 384 767 751 720

735 271 740 275 756

554 518 748 748 403

393 156 778 755 541

781 371 798 664 530

321 640 461 706 421

10 193 611 386 800

164 53 195 566 702

117 555 203 630 578

777 2 792 24 159

757 736 786 765 455

237 152 519 690 528

748 738 792 795 566

43 593 610 645 713

674 444 690 800 347

158 526 780 696 770

359 93 720 538 311

Out #7 :

1826794736

In #8 :

100

403 259 535 396 944

38 919 265 949 714

90 514 801 818 229

348 725 370 944 991

578 475 850 628 161

458 759 703 933 80

279 718 510 769 493

898 828 940 880 714

336 991 656 995 335

29 749 943 954 644

452 508 665 713 251

715 247 867 885 132

393 93 956 130 190

711 852 788 902 938

941 571 961 615 362

989 851 996 903 124

613 214 785 685 655

994 171 1000 940 563

202 52 434 265 895

19 922 837 970 786

606 129 979 234 995

143 441 711 742 944

731 489 904 780 27

860 617 930 763 437

194 879 962 933 795

556 612 676 740 405

512 135 836 750 406

803 396 980 494 442

980 702 997 744 977

885 59 912 672 344

60 911 809 974 800

742 328 747 524 602

327 380 412 778 76

832 99 890 115 782

863 457 989 590 430

746 50 776 452 342

641 461 914 462 757

524 584 582 943 125

70 952 151 982 348

37 666 235 727 525

269 441 858 578 387

272 214 832 509 733

320 693 647 784 500

621 617 734 704 200

915 866 972 973 20

922 381 966 1000 371

106 567 677 730 824

28 455 165 880 512

693 550 743 744 556

633 832 703 971 133

695 716 760 808 7

139 512 567 982 79

599 995 876 997 60

509 689 656 959 538

461 651 667 904 757

53 833 672 999 776

344 18 650 960 661

778 115 779 542 485

206 318 506 877 713

452 927 482 997 953

679 617 821 847 583

839 445 924 633 496

235 292 947 782 361

491 247 754 317 451

319 846 444 920 8

91 951 484 997 84

919 603 930 646 826

576 721 882 827 589

291 891 313 927 817

475 491 896 949 492

887 757 938 937 253

232 421 270 486 55

843 830 954 981 728

172 845 255 868 978

535 855 818 996 358

262 825 623 880 585

96 700 771 807 72

129 904 591 918 982

267 446 508 784 708

857 972 878 994 787

956 28 965 882 717

977 154 981 608 754

465 253 618 400 331

817 599 951 902 743

326 531 866 684 263

466 514 835 725 396

513 341 697 372 64

74 708 262 839 999

653 625 780 792 210

244 295 609 427 53

405 578 992 946 999

541 283 845 451 812

675 536 849 829 446

158 727 771 899 27

768 55 999 598 333

294 220 577 691 652

749 988 912 994 431

208 682 977 799 665

344 635 919 806 847

260 209 538 944 227

Out #8 :

2609260075

In #9 :

100

504 11 834 129 16

695 101 836 810 875

905 200 976 770 168

424 665 575 820 59

199 738 818 751 559

919 940 991 956 683

535 272 940 615 951

985 445 999 458 109

253 36 548 651 351

841 512 878 635 604

413 940 529 990 239

132 921 501 936 272

688 959 956 961 508

890 313 990 360 286

935 508 948 569 763

971 347 983 742 194

334 510 623 635 41

521 571 653 728 603

95 353 923 414 868

899 233 956 584 87

142 928 570 937 627

805 575 825 606 802

672 775 746 882 604

966 144 995 846 999

83 652 911 712 288

43 913 969 948 557

518 389 966 687 768

693 249 810 287 782

526 577 835 867 525

899 185 941 254 317

462 458 876 719 521

5 571 672 656 824

656 929 906 946 276

524 513 573 997 885

929 279 950 988 624

686 377 762 965 618

90 458 932 799 665

592 650 876 994 406

897 230 986 288 504

869 626 962 751 224

498 425 801 650 568

452 960 991 962 610

390 104 401 893 550

577 878 950 940 417

188 769 923 963 372

315 308 936 428 186

617 682 846 809 650

671 528 736 919 122

661 57 872 903 528

723 281 947 702 368

140 202 898 980 39

605 394 846 637 70

610 857 939 936 241

918 694 921 814 650

929 442 945 584 460

86 32 934 181 19

225 699 306 726 293

847 419 915 726 423

883 110 980 664 496

940 974 980 977 347

785 165 936 486 618

575 947 718 965 563

47 155 723 520 311

759 988 933 991 78

349 921 496 937 315

762 702 896 826 915

158 519 556 697 482

818 673 866 692 708

181 535 979 872 271

717 797 930 829 740

11 662 300 873 562

395 350 945 725 760

808 525 916 636 186

557 666 596 926 927

139 361 550 981 366

80 68 363 293 658

593 117 852 683 324

917 886 973 973 726

912 443 919 477 577

769 796 962 831 105

742 24 935 191 529

574 285 837 696 373

564 401 951 998 865

19 750 602 956 945

558 572 645 790 757

525 936 653 990 237

116 814 344 891 384

762 687 787 964 1000

300 13 317 984 463

5 911 544 945 421

75 334 446 503 355

839 45 955 986 279

969 572 999 732 233

470 182 690 852 527

449 179 648 815 268

469 330 740 348 883

751 389 967 891 499

269 51 472 423 860

682 863 836 918 854

894 536 983 732 763

Out #9 :

2665563724

In #10 :

2

5 5 8 8 8

10 10 16 16 9

Out #10 :

0

\*/