## Question 1: BACTERIA - SPOJ Custom Test

This problem is to simulate "custom test" function in other Online Judges. The time limit for this problem is 30 seconds.

### Input

No specified input. The judge input file is actually an empty file.

### Output

Any output will be accepted. Click on "accepted" verdict to see the output of your program. The output limit for this problem is 1024KB. If your output exceeds 1024KB, only the first 1024KB will be displayed. Any output will receive 0 points as judge result.

Solution:

1. **using** System;
3. **public** **class** Test
4. {
5. **public** **static** **void** Main(**string**[] args)
6. {
7. Console.WriteLine("Hello World");
8. }
9. }

## Question 2: TESTINT - Test 1

Given two natural numbers (both not greater than 200), each number in the separate line, please print the sum of them.

### Example

**Input:**

2

3

**Output:**

5

Solution:

1. **using** System;
3. **class** Program
4. {
5. **static** **void** Main(**string**[] args)
6. {
7. **string** s1, s2;
9. **int** a1 = **int**.Parse(Console.ReadLine());
10. **int** a2 = **int**.Parse(Console.ReadLine());
12. Console.WriteLine(a1 + a2);
13. }
14. }

## Question 3. STRHH - Half of the half

[#basics](http://www.spoj.com/problems/tag/basics)

Given a sequence of 2\*k characters, please print every second character from the first half of the sequence. Start printing with the first character.

### Input

In the first line of input your are given the positive integer t (1<=t<=100) - the number of test cases. In the each of the next t lines, you are given a sequence of 2\*k (1<=k<=100) characters.

### Output

For each of the test cases please please print every second character from the first half of a given sequence (the first character should appear).

### Example

**Input:**

4

your

progress

is

noticeable

**Output:**

y

po

i

ntc

Solution:

1. **using** System;
2. **using** System.Collections.Generic;
3. **using** System.Linq;
4. **using** System.Text;
5. **using** System.Threading.Tasks;
7. **class** Program
8. {
9. **static** **void** Main(**string**[] args)
10. { **int** noofTestCases;
11. String input,temp;
12. List<String> output = **new** List<String>();
13. **do**
14. {
15. noofTestCases = **int**.Parse(Console.ReadLine());
16. } **while** (noofTestCases < 1 || noofTestCases > 100);
17. **for** (**int** i = 0; i < noofTestCases; i++)
18. {
19. **do**
20. {
21. input = Console.ReadLine();
22. } **while** (input.Length / 2 > 100 || input.Length / 2 < 1);
23. temp = String.Empty;
24. **for** (**int** j = 0; j < input.Length/2; j++)
25. {
26. **if** (j % 2 == 0 )
27. {
28. temp = temp + input[j];
29. }
30. }
31. output.Add(temp);
33. }
35. **foreach** (**object** opt **in** output) {
36. Console.WriteLine(opt);
37. }
38. }
39. }

## Question 4. CPTTRN1 - Character Patterns (Act 1)

[#basics](http://www.spoj.com/problems/tag/basics)

Using two characters: . (dot) and \* (asterisk) print a chessboard-like pattern. The first character printed should be \* (asterisk).

### Input

You are given t < 100 - the number of test cases and for each of the test cases two positive integers: l - the number of lines and c - the number of columns in the pattern (l, c < 100).

### Output

For each of the test cases output the requested pattern (please have a look at the example). Use one line break in between successive patterns.

### Example

1. Input:
2. 3
3. 3 1
4. 4 4
5. 2 5
7. Output:
8. \*
9. .
10. \*
12. \*.\*.
13. .\*.\*
14. \*.\*.
15. .\*.\*
17. \*.\*.\*
18. .\*.\*.

Solution:

1. **using** System;
2. **using** System.Collections.Generic;
3. **using** System.Linq;
4. **using** System.Text;
5. **using** System.Threading.Tasks;
7. **class** Program
8. {
9. **static** **void** Main(**string**[] args)
10. {
11. **string** input,snum1,snum2;
12. **int** noOfTestCases=0,num1=0,num2=0;
13. Boolean num1flag,num2flag;
14. List<**int**> inputListRow = **new** List<**int**>();
15. List<**int**> inputListColumn = **new** List<**int**>();
16. **do**
17. {
18. noOfTestCases = **int**.Parse(Console.ReadLine());
19. } **while** (noOfTestCases > 100||noOfTestCases<1);
21. **for** (**int** i = 0; i < noOfTestCases; i++)
22. {
23. **do**
24. {
25. input = String.Empty;
26. snum1 = String.Empty;
27. snum2 = String.Empty;
28. num1flag = **true**;
29. num2flag = **false**;
30. input = Console.ReadLine();
31. input = input.Trim();
32. **for** (**int** j = 0; j < input.Length; j++)
33. {
34. **if** (input[j] == ' ' && num1flag == **true**)
35. {
36. num1flag = **false**;
37. num2flag = **true**;
38. }
39. **if** (num1flag == **true** && input[j] != ' ')
40. {
41. snum1 = snum1 + input[j];
42. }
43. **if** (num2flag == **true** && input[j] != ' ')
44. {
45. snum2 = snum2 + input[j];
46. }
47. **if** (j == input.Length - 1)
48. {
49. num1 = **int**.Parse(snum1);
50. num2 = **int**.Parse(snum2);
52. }
53. }
54. } **while** (num1 > 100 || num2 > 100);
56. inputListRow.Add(num1);
57. inputListColumn.Add(num2);
58. }

61. **for** (**int** i = 0; i < inputListRow.Count; i++)
62. {   **for** (**int** j = 0; j < inputListColumn.Count; j++)
63. {
64. **for** (**int** k=0;k<inputListRow[i];k++)
65. {   **for** (**int** p = 0; p < inputListColumn[j]; p++)
66. {   **if**(k % 2 == 0 && p % 2 == 0)
67. Console.Write("\*");
68. **if** (k % 2 == 0 && p % 2 == 1)
69. Console.Write(".");
70. **if** (k % 2 == 1 && p % 2 == 0)
71. Console.Write(".");
72. **if** (k % 2 == 1 && p % 2 == 1)
73. Console.Write("\*");
74. }
75. Console.WriteLine();
77. }
78. i++;
79. Console.WriteLine();
80. }
81. **break**;
82. }
83. }
84. }

## Question 5. CPTTRN2 - Character Patterns (Act 2)

[#basics](http://www.spoj.com/problems/tag/basics)

Using two characters: . (dot) and \* (asterisk) print a frame-like pattern.

### Input

You are given t - the number of test cases and for each of the test cases two positive integers: l - the number of lines and c - the number of columns of a frame.

### Output

For each of the test cases output the requested pattern (please have a look at the example). Use one line break in between successive patterns.

### Example

1. Input:
2. 3
3. 3 1
4. 4 4
5. 2 5
7. Output:
8. \*
9. \*
10. \*
12. \*\*\*\*
13. \*..\*
14. \*..\*
15. \*\*\*\*

18. \*\*\*\*\*
19. \*\*\*\*\*

Solution:

1. **using** System;
2. **using** System.Collections.Generic;
4. **class** Program
5. {
6. **static** **void** Main(**string**[] args)
7. {
8. **string** input, snum1, snum2;
9. **int** noOfTestCases = 0, num1 = 0, num2 = 0;
10. Boolean num1flag, num2flag;
11. List<**int**> inputListRow = **new** List<**int**>();
12. List<**int**> inputListColumn = **new** List<**int**>();
13. **do**
14. {
15. noOfTestCases = **int**.Parse(Console.ReadLine());
16. } **while** (noOfTestCases > 100 || noOfTestCases < 1);
18. **for** (**int** i = 0; i < noOfTestCases; i++)
19. {
20. **do**
21. {
22. input = String.Empty;
23. snum1 = String.Empty;
24. snum2 = String.Empty;
25. num1flag = **true**;
26. num2flag = **false**;
27. input = Console.ReadLine();
28. input = input.Trim();
29. **for** (**int** j = 0; j < input.Length; j++)
30. {
31. **if** (input[j] == ' ' && num1flag == **true**)
32. {
33. num1flag = **false**;
34. num2flag = **true**;
35. }
36. **if** (num1flag == **true** && input[j] != ' ')
37. {
38. snum1 = snum1 + input[j];
39. }
40. **if** (num2flag == **true** && input[j] != ' ')
41. {
42. snum2 = snum2 + input[j];
43. }
44. **if** (j == input.Length - 1)
45. {
46. num1 = **int**.Parse(snum1);
47. num2 = **int**.Parse(snum2);
49. }
50. }
51. } **while** (num1 > 100 || num2 > 100);
53. inputListRow.Add(num1);
54. inputListColumn.Add(num2);
55. }

58. **for** (**int** i = 0; i < inputListRow.Count; i++)
59. {
60. **for** (**int** j = 0; j < inputListColumn.Count; j++)
61. {
62. **for** (**int** k = 0; k < inputListRow[i]; k++)
63. {
64. **for** (**int** p = 0; p < inputListColumn[j]; p++)
65. {   **if** (k == 0 || k == inputListRow[i]-1)
66. Console.Write("\*");
67. **else** **if** (p == 0 || p == inputListColumn[j]-1)
68. Console.Write("\*");
69. **else**
70. Console.Write(".");
72. }
73. Console.WriteLine();
75. }
76. i++;
77. Console.WriteLine();
78. }
79. **break**;
80. }
81. }
82. }

## Question 6. CPTTRN3 - Character Patterns (Act 3)

[#basics](http://www.spoj.com/problems/tag/basics)

Using two characters: . (dot) and \* (asterisk) print a grid-like pattern.

### Input

You are given t - the number of test cases and for each of the test cases two positive integers: l - the number of lines and c - the number of columns in the grid. Each square of the grid is of the same size and filled with 4 dots (see the example below).

### Output

For each of the test cases output the requested pattern (please have a look at the example). Use one line break in between successive patterns.

### Example

1. Input:
2. 3
3. 3 1
4. 4 4
5. 2 5
7. Output:
8. \*\*\*\*
9. \*..\*
10. \*..\*
11. \*\*\*\*
12. \*..\*
13. \*..\*
14. \*\*\*\*
15. \*..\*
16. \*..\*
17. \*\*\*\*
19. \*\*\*\*\*\*\*\*\*\*\*\*\*
20. \*..\*..\*..\*..\*
21. \*..\*..\*..\*..\*
22. \*\*\*\*\*\*\*\*\*\*\*\*\*
23. \*..\*..\*..\*..\*
24. \*..\*..\*..\*..\*
25. \*\*\*\*\*\*\*\*\*\*\*\*\*
26. \*..\*..\*..\*..\*
27. \*..\*..\*..\*..\*
28. \*\*\*\*\*\*\*\*\*\*\*\*\*
29. \*..\*..\*..\*..\*
30. \*..\*..\*..\*..\*
31. \*\*\*\*\*\*\*\*\*\*\*\*\*
33. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
34. \*..\*..\*..\*..\*..\*
35. \*..\*..\*..\*..\*..\*
36. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
37. \*..\*..\*..\*..\*..\*
38. \*..\*..\*..\*..\*..\*
39. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Solution:

1. **using** System;
2. **using** System.Collections.Generic;
4. **class** Program
5. {
6. **static** **void** Main(**string**[] args)
7. {
8. **string** input, snum1, snum2;
9. **int** noOfTestCases = 0, num1 = 0, num2 = 0;
10. Boolean num1flag, num2flag;
11. List<**int**> inputListRow = **new** List<**int**>();
12. List<**int**> inputListColumn = **new** List<**int**>();
13. **do**
14. {
15. noOfTestCases = **int**.Parse(Console.ReadLine());
16. } **while** (noOfTestCases > 100 || noOfTestCases < 1);
18. **for** (**int** i = 0; i < noOfTestCases; i++)
19. {
20. **do**
21. {
22. input = String.Empty;
23. snum1 = String.Empty;
24. snum2 = String.Empty;
25. num1flag = **true**;
26. num2flag = **false**;
27. input = Console.ReadLine();
28. input = input.Trim();
29. **for** (**int** j = 0; j < input.Length; j++)
30. {
31. **if** (input[j] == ' ' && num1flag == **true**)
32. {
33. num1flag = **false**;
34. num2flag = **true**;
35. }
36. **if** (num1flag == **true** && input[j] != ' ')
37. {
38. snum1 = snum1 + input[j];
39. }
40. **if** (num2flag == **true** && input[j] != ' ')
41. {
42. snum2 = snum2 + input[j];
43. }
44. **if** (j == input.Length - 1)
45. {
46. num1 = **int**.Parse(snum1);
47. num2 = **int**.Parse(snum2);
49. }
50. }
51. } **while** (num1 > 100 || num2 > 100);
53. inputListRow.Add(num1);
54. inputListColumn.Add(num2);
55. }

58. **int** x = 0;
59. **for** (**int** j = 0; j < inputListColumn.Count; j++)
60. {
61. functionprintsquare(inputListRow[x], inputListColumn[j]);
62. **if**(x<inputListRow.Count)
63. x++;
64. Console.WriteLine();
65. }



70. }
72. **public** **static** **void** functionprintsquare(**int** rowListi, **int** columnListj)
73. {
74. **int** x;
75. **bool** aflag=**true**, bflag=**false**;
77. **for** (**int** y = 0; y < 4 \* rowListi; y++)
78. {
79. aflag = **true**;
80. bflag = **false**;
81. **if** (y % 4 == 0 && y/4==0 || y % 4 == 3)
82. {
83. **for** (**int** z = 0; z < 4 \* columnListj-columnListj+1; z++)
84. {
85. Console.Write("\*");
86. **if** (z == 4 \* columnListj - columnListj)
87. {
88. Console.WriteLine();
89. }
90. }
92. }
93. **else** **if** (y % 4 == 1 || y % 4 == 2)
94. {
95. x = 0;
96. **while**(x<4\*columnListj-columnListj+1)
97. {   **if** (aflag == **true**)
98. {
99. Console.Write("\*..\*");
100. x = x + 4;
101. aflag = **false**;
102. bflag = **true**;
103. }
104. **if** (bflag == **true** && x!=4\*columnListj-columnListj+1)
105. {
106. Console.Write("..\*");
107. x = x + 3;
108. }
109. **if** (x == 4 \* columnListj - columnListj+1)
110. {
111. Console.WriteLine();
112. }
114. }
116. }
117. }
118. }
119. }

## Question 7. CPTTRN5 - Character Patterns (Act 5)

[#basics](http://www.spoj.com/problems/tag/basics)

Using two characters: . (dot) and \* (asterisk) print a grid-like pattern. The grid will have l lines, c columns, and each square shaped element of the grid will have the height and width equal to s.

Moreover, each of the grid elements will have a diagonal. The diagonal of the first square in the first line of the grid is directed towards down and right corner - use the \ (backslash) character to print it; while the next diagonal will be directed towards upper right corner - use the / (slash) character to print it. Print the successive diagonals alternately (please consult the example below).

### Input

You are given t - the number of test cases and for each of the test case three positive integers: l - the number of lines, c - the number of columns in the grid and s - the size of the single square shaped element.

### Output

For each of the test cases output the requested pattern (please have a look at the example). Use one line break in between successive patterns.

### Example

1. Input:
2. 3
3. 3 1 2
4. 4 4 1
5. 2 5 2
7. Output:
8. \*\*\*\*
9. \*\.\*
10. \*.\\*
11. \*\*\*\*
12. \*./\*
13. \*/.\*
14. \*\*\*\*
15. \*\.\*
16. \*.\\*
17. \*\*\*\*
19. \*\*\*\*\*\*\*\*\*
20. \*\\*/\*\\*/\*
21. \*\*\*\*\*\*\*\*\*
22. \*/\*\\*/\*\\*
23. \*\*\*\*\*\*\*\*\*
24. \*\\*/\*\\*/\*
25. \*\*\*\*\*\*\*\*\*
26. \*/\*\\*/\*\\*
27. \*\*\*\*\*\*\*\*\*
29. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
30. \*\.\*./\*\.\*./\*\.\*
31. \*.\\*/.\*.\\*/.\*.\\*
32. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
33. \*./\*\.\*./\*\.\*./\*
34. \*/.\*.\\*/.\*.\\*/.\*
35. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Solution:

1. **using** System;
2. **using** System.Collections.Generic;
3. **using** System.Linq;
4. **class** Program
5. {   **public** **static** **void** FunctionPrintSquares(**int** l, **int** c, **int** s)
6. {   **for** (**int** i = 0; i < (l \* s + l + 1); i++)
7. {   **for** (**int** j = 0; j < (c \* s + c + 1); j++)
8. {
9. **if** ((i - 1) % (s + 1) == -1)
10. {
11. Console.Write("\*");
12. }
13. **else** **if** ((j - 1) % (s + 1) == -1)
14. {
15. Console.Write("\*");
16. }
17. **else** **if** ((i - 1) % (s + 1) == s)
18. {
19. Console.Write("\*");
20. }
21. **else** **if** ((j - 1) % (s + 1) == s)
22. {
23. Console.Write("\*");
24. }
25. **else** **if** ((i - 1) % (s + 1) != s && (j - 1) % (s + 1) != s && ((i - 1) + (j - 1)) % 2 == 0)
26. {
27. **if** (((i - 1) / (s + 1)) % 2 == 0 && ((j - 1) / (s + 1)) % 2 == 0)
28. Console.Write("\\");
29. **else** **if** (((i - 1) / (s + 1)) % 2 == 1 && ((j - 1) / (s + 1)) % 2 == 0)
30. Console.Write("/");
31. **else** **if** (((i - 1) / (s + 1)) % 2 == 0 && ((j - 1) / (s + 1)) % 2 == 1)
32. {
33. Console.Write("/");
34. }
35. **else** **if** (((i - 1) / (s + 1)) % 2 == 1 && ((j - 1) / (s + 1)) % 2 == 1)
36. {
37. Console.Write("\\");
38. }
40. }
41. **else** **if** ((i - 1) % (s + 1) != s && (j - 1) % (s + 1) != s && ((i - 1) + (j - 1)) % 2 == 1)
42. {
43. Console.Write(".");
44. }
45. }
46. Console.WriteLine();
47. }
48. }
49. **static** **void** Main(**string**[] args)
50. {
51. **string** input, snum1, snum2, snum3;
52. **int** noOfTestCases = 0, num1 = 0, num2 = 0, num3=0;
53. Boolean num1flag, num2flag, num3flag;
54. List<**int**> inputListLines = **new** List<**int**>();
55. List<**int**> inputListColumn = **new** List<**int**>();
56. List<**int**> inputListSize = **new** List<**int**>();
57. **do**
58. {
59. noOfTestCases = **int**.Parse(Console.ReadLine());
60. } **while** (noOfTestCases < 1);
62. **for** (**int** i = 0; i < noOfTestCases; i++)
63. {
64. **do**
65. {
66. input = String.Empty;
67. snum1 = String.Empty;
68. snum2 = String.Empty;
69. snum3 = String.Empty;
70. num1flag = **true**;
71. num2flag = **false**;
72. num3flag = **false**;
73. input = Console.ReadLine();
74. input = input.Trim();
75. **for** (**int** j = 0; j < input.Length; j++)
76. {
77. **if** (input[j] == ' ')
78. {
79. **if** (num1flag == **true**)
80. {
81. num1flag = **false**;
82. num2flag = **true**;
83. }
84. **else** **if** (num2flag == **true**)
85. {
86. num2flag = **false**;
87. num3flag = **true**;
88. }
89. }
90. **if** (num1flag == **true** && input[j] != ' ')
91. {
92. snum1 = snum1 + input[j];
93. }
94. **if** (num2flag == **true** && input[j] != ' ')
95. {
96. snum2 = snum2 + input[j];
97. }
98. **if** (num3flag == **true** && input[j] != ' ')
99. {
100. snum3 = snum3 + input[j];
101. }
102. **if** (j == input.Length - 1)
103. {
104. num1 = **int**.Parse(snum1);
105. num2 = **int**.Parse(snum2);
106. num3 = **int**.Parse(snum3);
108. }
109. }
110. } **while** (num1 <1 || num2 < 1||num3<1);
112. inputListLines.Add(num1);
113. inputListColumn.Add(num2);
114. inputListSize.Add(num3);
116. }
118. **for** (**int** i = 0; i < inputListLines.Count(); i++)
119. {
120. FunctionPrintSquares(inputListLines[i], inputListColumn[i], inputListSize[i]);
121. Console.WriteLine();
122. }
123. }
124. }

## Question 8. CPTTRN6 - Character Patterns (Act 6)

[#basics](http://www.spoj.com/problems/tag/basics)

Given specified dimensions, print a grid-like pattern. Use the | (pipe) sign to print vertical elements, the - (minus) to print horizontal ones and + (plus) for crossings. The rest of the space fill with . (dots) characters.

### Input

You are given t - the number of test cases and for each of the test cases four positive integers: l - the number of horizontal elements, c - the number of vertical elements in the grid; h and w - the high and the with of the single rectangle respectively.

### Output

For each of the test cases output the requested pattern (please have a look at the example). Use one line break in between successive patterns.

### Example

1. Input:
2. 3
3. 3 1 2 1
4. 4 4 1 2
5. 2 5 3 2
7. Output:
8. .|.
9. .|.
10. -+-
11. .|.
12. .|.
13. -+-
14. .|.
15. .|.
16. -+-
17. .|.
18. .|.
20. ..|..|..|..|..
21. --+--+--+--+--
22. ..|..|..|..|..
23. --+--+--+--+--
24. ..|..|..|..|..
25. --+--+--+--+--
26. ..|..|..|..|..
27. --+--+--+--+--
28. ..|..|..|..|..

31. ..|..|..|..|..|..
32. ..|..|..|..|..|..
33. ..|..|..|..|..|..
34. --+--+--+--+--+--
35. ..|..|..|..|..|..
36. ..|..|..|..|..|..
37. ..|..|..|..|..|..
38. --+--+--+--+--+--
39. ..|..|..|..|..|..
40. ..|..|..|..|..|..
41. ..|..|..|..|..|..

Solution:

1. **using** System;
2. **using** System.Collections.Generic;
3. **using** System.Linq;

6. **class** Program
7. {   **public** **static** **void** functionprintpattern(**int** r, **int** c, **int** h, **int** w)
8. {   **for** (**int** i = 0; i < (h \* (r + 1) + r); i++)
9. {   **for** (**int** j = 0; j < (w \* (c + 1) + c); j++)
10. {
11. **if** (i % (h + 1) == h)
12. {
13. **if** (j % (w+1) == w )
14. {
15. Console.Write("+");
16. }
17. **else**
18. {
19. Console.Write("-");
20. }
21. }
22. **else**
23. {
24. **if** (j % (w+1) == w )
25. {
26. Console.Write("|");
27. }
28. **else**
29. {
30. Console.Write(".");
31. }
32. }
33. }
34. Console.WriteLine();
35. }
36. }
37. **static** **void** Main(**string**[] args)
38. {
39. **string** input, snum1, snum2, snum3, snum4;
40. **int** noOfTestCases = 0, num1 = 0, num2 = 0, num3 = 0, num4 = 0;
41. Boolean num1flag, num2flag, num3flag, num4flag;
42. List<**int**> inputListrow = **new** List<**int**>();
43. List<**int**> inputListcolumn = **new** List<**int**>();
44. List<**int**> inputListHeight = **new** List<**int**>();
45. List<**int**> inputListWidth = **new** List<**int**>();
47. **do**
48. {
49. noOfTestCases = **int**.Parse(Console.ReadLine());
50. } **while** (noOfTestCases > 100 || noOfTestCases < 1);
52. **for** (**int** i = 0; i < noOfTestCases; i++)
53. {
54. **do**
55. {
56. input = String.Empty;
57. snum1 = String.Empty;
58. snum2 = String.Empty;
59. snum3 = String.Empty;
60. snum4 = String.Empty;
62. num1flag = **true**;
63. num2flag = **false**;
64. num3flag = **false**;
65. num4flag = **false**;
66. input = Console.ReadLine();
67. input = input.Trim();
68. **for** (**int** j = 0; j < input.Length; j++)
69. {
70. **if** (input[j] == ' ')
71. {
72. **if** (num1flag == **true**)
73. {
74. num1flag = **false**;
75. num2flag = **true**;
76. }
77. **else** **if** (num2flag == **true**)
78. {
79. num2flag = **false**;
80. num3flag = **true**;
81. }
82. **else** **if** (num3flag == **true**)
83. {
84. num3flag = **false**;
85. num4flag = **true**;
86. }
87. }
88. **if** (num1flag == **true** && input[j] != ' ')
89. {
90. snum1 = snum1 + input[j];
91. }
92. **if** (num2flag == **true** && input[j] != ' ')
93. {
94. snum2 = snum2 + input[j];
95. }
96. **if** (num3flag == **true** && input[j] != ' ')
97. {
98. snum3 = snum3 + input[j];
99. }
100. **if** (num4flag == **true** && input[j] != ' ')
101. {
102. snum4 = snum4 + input[j];
103. }
104. **if** (j == input.Length - 1)
105. {
106. num1 = **int**.Parse(snum1);
107. num2 = **int**.Parse(snum2);
108. num3 = **int**.Parse(snum3);
109. num4 = **int**.Parse(snum4);
110. }
111. }
112. } **while** (num1 < 1 || num2 < 1||num3 <1||num4<1);
114. inputListrow.Add(num1);
115. inputListcolumn.Add(num2);
116. inputListHeight.Add(num3);
117. inputListWidth.Add(num4);
118. }
120. **for** (**int** i = 0; i < inputListrow.Count(); i++)
121. {
122. functionprintpattern(inputListrow[i], inputListcolumn[i], inputListHeight[i], inputListWidth[i]);
123. Console.WriteLine();
124. }
125. }
126. }

## CPTTRN7 - Character Patterns (Act 7)

Print a regular grid pattern with diamond-like base elements. Use the \ (backslash) and the / (slash) characters to print diamonds and . (dots) to fill the rest of the space.

Input

You are given t - the number of test cases and for each of the test cases three positive integers: r - the number of rows, c - the number of columns in the grid and s - the size of each diamond.

### Output

For each of the test cases output the requested pattern (please have a look at the example). Use one line break in between successive patterns.

Example

1. Input:
2. 3
3. 3 1 2
4. 4 4 1
5. 2 5 2
7. Output:
8. ./\.
9. /..\
10. \../
11. .\/.
12. ./\.
13. /..\
14. \../
15. .\/.
16. ./\.
17. /..\
18. \../
19. .\/.
21. /\/\/\/\
22. \/\/\/\/
23. /\/\/\/\
24. \/\/\/\/
25. /\/\/\/\
26. \/\/\/\/
27. /\/\/\/\
28. \/\/\/\/

31. ./\../\../\../\../\.
32. /..\/..\/..\/..\/..\
33. \../\../\../\../\../
34. .\/..\/..\/..\/..\/.
35. ./\../\../\../\../\.
36. /..\/..\/..\/..\/..\
37. \../\../\../\../\../
38. .\/..\/..\/..\/..\/.

Solution:

1. **using** System;
2. **using** System.Collections.Generic;
3. **using** System.Linq;
5. **class** Program
6. {   **public** **static** **void** FuncPrintPattern(**int** r, **int** c, **int** s)
7. {   **for** (**int** i = 0; i < (s \* r\*2); i++)
8. {   **if** (i % (s \* 2) == 0)
9. {
10. FuncPrintFirstLine(c, s);
11. }
12. **else** **if** (i % (s \* 2) < s && i % (s \* 2) != 0)
13. {
14. FuncPrintUpperLine(i, c, s);
15. }
16. **else** **if** (i % (s \* 2) >= s && i % (s \* 2) != (s\*2 - 1))
17. {
18. FuncPrintLowerLine(i, c, s);
19. }
20. **else** **if** (i % (s \* 2) == (s\*2 - 1))
21. {
22. FuncPrintLastLine(c, s);
23. }
24. Console.WriteLine();
26. }
27. }
29. **public** **static** **void** FuncPrintUpperLine(**int** nthLine, **int** col, **int** size)
30. {   **for** (**int** j = 0; j < (size \* col \* 2); j++)
31. {   **if** (j % size == (size - 1 - (nthLine % size)) && (j/size)%2==0)
32. Console.Write("/");
33. **else** **if** (j % size == (nthLine % size) && (j / size) % 2 == 1)
34. Console.Write("\\");
35. **else**
36. Console.Write(".");
37. }
38. }
40. **public** **static** **void** FuncPrintLowerLine(**int** nthLine, **int** col, **int** size)
41. {   **for** (**int** j = 0; j < (size \* col \* 2); j++)
42. {   **if** (j % size == (nthLine%size) && (j/size)%2==0)
43. Console.Write("\\");
44. **else** **if** (j % size == (size - 1 - (nthLine%size)) && (j / size) % 2 == 1)
45. Console.Write("/");
46. **else**
47. Console.Write(".");
49. }
51. }
53. **public** **static** **void** FuncPrintLastLine(**int** col, **int** size)
54. {   **for** (**int** j = 0; j < (size \* col \* 2); j++)
55. {   **if** (j % size == size - 1 && (j/size)%2==0)
56. Console.Write("\\");
57. **else** **if** (j % size == 0 && (j / size) % 2 == 1)
58. Console.Write("/");
59. **else**
60. Console.Write(".");
61. }
62. }
64. **public** **static** **void** FuncPrintFirstLine(**int** col, **int** size)
65. {   **for** (**int** j = 0; j < (size \* col\*2); j++)
66. {   **if** (j % size == (size - 1)&&(j/size)%2==0)
67. Console.Write("/");
68. **else** **if** (j % size == 0 && (j / size)%2 == 1 )
69. Console.Write("\\");
70. **else**
71. Console.Write(".");
72. }
73. }
74. **static** **void** Main(**string**[] args)
75. {
76. **string** input, snum1, snum2, snum3;
77. **int** noOfTestCases = 0, num1 = 0, num2 = 0, num3 = 0;
78. Boolean num1flag, num2flag, num3flag;
79. List<**int**> inputListRows = **new** List<**int**>();
80. List<**int**> inputListColumn = **new** List<**int**>();
81. List<**int**> inputListSize = **new** List<**int**>();
82. **do**
83. {
84. noOfTestCases = **int**.Parse(Console.ReadLine());
85. } **while** (noOfTestCases < 1);
87. **for** (**int** i = 0; i < noOfTestCases; i++)
88. {
89. **do**
90. {
91. input = String.Empty;
92. snum1 = String.Empty;
93. snum2 = String.Empty;
94. snum3 = String.Empty;
95. num1flag = **true**;
96. num2flag = **false**;
97. num3flag = **false**;
98. input = Console.ReadLine();
99. input = input.Trim();
100. **for** (**int** j = 0; j < input.Length; j++)
101. {
102. **if** (input[j] == ' ')
103. {
104. **if** (num1flag == **true**)
105. {
106. num1flag = **false**;
107. num2flag = **true**;
108. }
109. **else** **if** (num2flag == **true**)
110. {
111. num2flag = **false**;
112. num3flag = **true**;
113. }
114. }
115. **if** (num1flag == **true** && input[j] != ' ')
116. {
117. snum1 = snum1 + input[j];
118. }
119. **if** (num2flag == **true** && input[j] != ' ')
120. {
121. snum2 = snum2 + input[j];
122. }
123. **if** (num3flag == **true** && input[j] != ' ')
124. {
125. snum3 = snum3 + input[j];
126. }
127. **if** (j == input.Length - 1)
128. {
129. num1 = **int**.Parse(snum1);
130. num2 = **int**.Parse(snum2);
131. num3 = **int**.Parse(snum3);
133. }
134. }
135. } **while** (num1 < 1 || num2 < 1 || num3 < 1);
137. inputListRows.Add(num1);
138. inputListColumn.Add(num2);
139. inputListSize.Add(num3);

142. }
144. **for** (**int** k = 0; k < inputListRows.Count(); k++)
145. {
146. FuncPrintPattern(inputListRows[k], inputListColumn[k], inputListSize[k]);
147. Console.WriteLine();
148. }
149. }
150. }