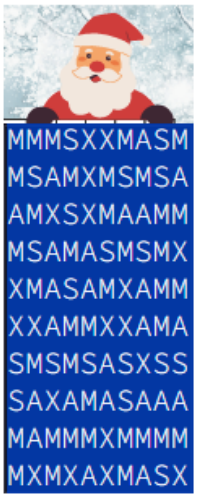
Challenge Jan-2025
Christmas Word Search

A solution with DT5GL/ Python
by Jack Jansonius – 05 January 2025

Problem Statement (from the website):

You need to find how many times the word "XMAS" appears in the grid, horizontally, vertically, or diagonally, written backwards, or even overlapping other words in any orientation. For example, you may find that XMAS occurs a total of *18* times in the grid on the right. You can find this grid here. Can you create a service capable of dealing with any grid?



Introduction
The grid consists of 10 lines, numbered from 0 to 9.
Each line consists of 10 characters, also numbered from 0 to 9.

So each character in the grid has a coordinate (line_nr, char_nr), so that (0,0) is the first character on the first line and (9,9) is the last character on the last line.

In pseudocode, the algorithm looks simple:

```
For line_nr from 0 to 9 do:
        For character_nr from 0 to 9 do:
                If character at coordinate (line_nr, character_nr) =
                    first character of searched word:
                Then:
                    For each direction, check:
                        - If direction in respect to space possible in grid:
                                check searched word in that direction
                                - If searched word present:
                                      report coordinate of word and direction.

                Else:
                    Do nothing (= skip character)
```

ChatGPT (but also Copilot in Jacob Feldman's first solution) comes up with a very ingenious notation for the direction to investigate:

```
directions = [
      (0, 1),   # Right
      (1, 0),   # Down
      (1, 1),   # Down-Right diagonal
      (1, -1),  # Down-Left diagonal
      (0, -1),  # Left
      (-1, 0),  # Up
      (-1, -1), # Up-Left diagonal
      (-1, 1)   # Up-Right diagonal ]
```

(0,1) then means: line_nr + 0, char_nr + 1, so 1 character to the right on the same line. And (-1, -1) means: line_nr -1, char_nr -1, so 1 line up and 1 character to the left in the grid.

Based on the length of the searched word, we can now check for each direction to be examined whether it makes sense to check whether the searched word is present.

For a certain coordinate in the grid for the direction DOWN only 1 condition is relevant, namely: line_nr + word_length <= grid_height

And for the direction RIGHT, also only 1 condition is relevant:
char_nr + word_length <= grid_width

And both of the above conditions are relevant for the DOWN-RIGHT direction.

In this way, for all 8 directions at a point in the grid (which must be the first letter of the searched word), only 1 or 2 conditions are always relevant.

Implementation of the decision model in DT5GL/Python:

```
Initial_instructions:
>>: grid_height = get_grid_height()           # from DTFunctions.py
>>: grid_width  = get_grid_width()
>>: lastLine = grid_height - 1
>>: lastChar = grid_width - 1
>>: word_to_find = "XMAS"
>>: word_length = len(word_to_find)
>>: total_found = 0
>>: True = 1
End_Instructions


rTable 0: Read next line
If:                                          | 0| 1|
next line_nr in [0 - lastLine]               | Y| N|
Then:
Line is selected                             | X|  |
Line is finished                             |  | X|
# .......

rTable 1: Read next character on line
If:                                          | 0| 1|
next char_nr in [0 - lastChar]               | Y| N|
Then:
Char is selected                             | X|  |
Char is finished                             |  | X|
# .......


rTable 2: Skip if the current position is not the first letter of the word
If:                                                | 0|
True = fgrid(line_nr,char_nr) != word_to_find[0]   | Y|
Then:
Direction is skipped                               | X|
# .......


# Direction starting up-left and going clockwise....

rTable 3: UP-LEFT, UP, UP-RIGHT[1]
If:                                                          | 0| 1| 2|
True = line_nr - word_length >= -1                           | Y| Y| Y|
True = char_nr - word_length >= -1                           | Y| -| -|
True = char_nr + word_length <= grid_width                   | -| -| Y|
True = is_word_at(line_nr, char_nr, -1, -1, word_to_find)    | Y| -| -|
True = is_word_at(line_nr, char_nr, -1,  0, word_to_find)    | -| Y| -|
True = is_word_at(line_nr, char_nr, -1,  1, word_to_find)    | -| -| Y|
Then:
Direction is UP-LEFT                                         | X|  |  |
Direction is UP                                              |  | X|  |
Direction is UP-RIGHT                                        |  |  | X|
# .......

rTable 4: RIGHT
If:                                                          | 0|
True = char_nr + word_length <= grid_width                   | Y|
True = is_word_at(line_nr, char_nr,  0,  1, word_to_find)    | Y|
Then:
Direction is RIGHT                                           | X|
# .......
```

[1] The 8 directions can also be included in one large table or in 8 separate tables.

```
rTable 5: DOWN-RIGHT, DOWN, DOWN-LEFT
If:                                                          | 0| 1| 2|
True = line_nr + word_length <= grid_height                  | Y| Y| Y|
True = char_nr + word_length <= grid_width                   | Y| -| -|
True = char_nr - word_length >= -1                           | -| -| Y|
True = is_word_at(line_nr, char_nr,  1,  1, word_to_find)    | Y| -| -|
True = is_word_at(line_nr, char_nr,  1,  0, word_to_find)    | -| Y| -|
True = is_word_at(line_nr, char_nr,  1, -1, word_to_find)    | -| -| Y|
Then:
Direction is DOWN-RIGHT                                      | X|  |  |
Direction is DOWN                                            |  | X|  |
Direction is DOWN-LEFT                                       |  |  | X|
# .......


rTable 6: LEFT
If:                                                          | 0|
True = char_nr - word_length >= -1                           | Y|
True = is_word_at(line_nr, char_nr,  0, -1, word_to_find)    | Y|
Then:
Direction is LEFT                                            | X|
# .......


Attribute: grid_height   Type: Integer
Attribute: grid_width    Type: Integer
Attribute: lastLine      Type: Integer
Attribute: lastChar      Type: Integer
Attribute: word_length   Type: Integer


GOALATTRIBUTE: Line
Repeat_until: finished

Case: finished
Print: "Total Words Found: %s" total_found

Case: selected
Print: "#REM# - "


GOALATTRIBUTE: Char
Repeat_until: finished

Case: finished
Print: "#REM# - "

Case: selected
Print: "#REM# - "
```

```
GOALATTRIBUTE: Direction
Multivalued_until: skipped

Case: skipped
    Print: "#REM# - "

Case: UP
    Print: "Start: (%s, %s), Direction: (-1, 0) (Up) "    line_nr  char_nr
    >>: total_found = total_found + 1
Case: UP-RIGHT
    Print: "Start: (%s, %s), Direction: (-1, 1) (Up-Right diagonal) "   line_nr
    char_nr
    >>: total_found = total_found + 1
Case: RIGHT
    Print: "Start: (%s, %s), Direction: (0, 1) (Right) "   line_nr  char_nr
    >>: total_found = total_found + 1
Case: DOWN-RIGHT
    Print: "Start: (%s, %s), Direction: (1, 1) (Down-Right diagonal) "  line_nr
    char_nr
    >>: total_found = total_found + 1
Case: DOWN
    Print: "Start: (%s, %s), Direction: (1, 0) (Down) "    line_nr  char_nr
    >>: total_found = total_found + 1
Case: DOWN-LEFT
    Print: "Start: (%s, %s), Direction: (1, -1) (Down-Left diagonal) "  line_nr
    char_nr
    >>: total_found = total_found + 1
Case: LEFT
    Print: "Start: (%s, %s), Direction: (0, -1) (Left) "   line_nr  char_nr
    >>: total_found = total_found + 1
Case: UP-LEFT
    Print: "Start: (%s, %s), Direction: (-1, -1) (Up-Left diagonal) "   line_nr
    char_nr
    >>: total_found = total_found + 1
```

Added to DTFunctions.py:

```
grid = [
    "MMMSXXMASM",
    "MSAMXMSMSA",
    "AMXSXMAAMM",
    "MSAMASMSMX",
    "XMASAMXAMM",
    "XXAMMXXAMA",
    "SMSMSASXSS",
    "SAXAMASAAA",
    "MAMMMXMMMM",
    "MXMXAXMASX"]

def fgrid(x,y):
    return grid[x][y]     # character on coordinate (line_nr, character_nr).

def get_grid_height():
    return len(grid)      # number of lines in the grid.

def get_grid_width():
    return len(grid[0])   # number of characters on first line of the grid.

def is_word_at(x, y, dx, dy, word):
    """Check if the word exists at the given starting point and direction."""
    for i in range(len(word)):
        nx, ny = x + i * dx, y + i * dy
        if grid[nx][ny] != word[i]:
            return False
    return True
```

Testrun (word_to_find = "XMAS")

```
PS C:\Users\administrator\dt5gl> .\dt.exe -nti[2]
File to read...: xmas.txt
Start: (0, 4), Direction: (1, 1) (Down-Right diagonal)
Start: (0, 5), Direction: (0, 1) (Right)
Start: (1, 4), Direction: (0, -1) (Left)
Start: (3, 9), Direction: (1, 0) (Down)
Start: (3, 9), Direction: (1, -1) (Down-Left diagonal)
Start: (4, 0), Direction: (0, 1) (Right)
Start: (4, 6), Direction: (-1, 0) (Up)
Start: (4, 6), Direction: (0, -1) (Left)
Start: (5, 0), Direction: (-1, 1) (Up-Right diagonal)
Start: (5, 6), Direction: (-1, -1) (Up-Left diagonal)
Start: (9, 1), Direction: (-1, 1) (Up-Right diagonal)
Start: (9, 3), Direction: (-1, -1) (Up-Left diagonal)
Start: (9, 3), Direction: (-1, 1) (Up-Right diagonal)
Start: (9, 5), Direction: (-1, -1) (Up-Left diagonal)
Start: (9, 5), Direction: (-1, 1) (Up-Right diagonal)
Start: (9, 5), Direction: (0, 1) (Right)
Start: (9, 9), Direction: (-1, -1) (Up-Left diagonal)
Start: (9, 9), Direction: (-1, 0) (Up)
Total Words Found: 18
Time elapsed: 0:00:01.097705
```

Testrun (word_to_find = "XMASA")

```
PS C:\Users\administrator\dt5gl> .\dt.exe -nti
File to read...: xmas.txt
Start: (3, 9), Direction: (1, 0) (Down)
Start: (3, 9), Direction: (1, -1) (Down-Left diagonal)
Start: (4, 0), Direction: (0, 1) (Right)
Start: (4, 6), Direction: (0, -1) (Left)
Start: (5, 6), Direction: (-1, -1) (Up-Left diagonal)
Start: (9, 3), Direction: (-1, 1) (Up-Right diagonal)
Start: (9, 5), Direction: (-1, 1) (Up-Right diagonal)
Start: (9, 9), Direction: (-1, 0) (Up)
Total Words Found: 8
Time elapsed: 0:00:01.362062
```

[2] DT.exe is Python code, compiled to C, and runs directly under Windows (without pre-installation of Python). Download and all necessary files available at: https://github.com/JackJansonius/DT5GL