Due Date: 13.11.2022, 23:55

CENG 113
Homework #1

Drawing Cards

In this homework, draw two cards next to each other. The texts should be centered both vertically and horizontally on the cards. The center lines of the two cards must be aligned. See the example runs below for a better understanding. Red texts are the example user inputs.

Example Run 1 (Case 1: the most important case)

```
This program will draw two cards next to each other.
──────────────────────
Texts must not be empty.
Text of first card: CENG113
Text of second card: Python
──────────────────────
Width of first card must be at least 9.
Width of first card: 11
Width of second card must be at least 8.
Width of second card: 11
──────────────────────
Heights must be odd and at least 3.
Height of first card: 5
Height of second card: 5
──────────────────────
┌─────────┬─────────┐
│ CENG113 │ Python  │
└─────────┴─────────┘
```

Let's annotate the lines, replace spaces around texts with '**x**'s, and replace border characters with '**+**'s to make it easier to understand:

```
++++++++++++++++++++  → Card 1 line 1, card 2 line 1
+xxxxxxxxx+xxxxxxxxx+  → Card 1 line 2, card 2 line 2
+xCENG113x+xPythonxx+  → Card 1 line 3, card 2 line 3
+xxxxxxxxx+xxxxxxxxx+  → Card 1 line 4, card 2 line 4
++++++++++++++++++++  → Card 1 line 5, card 2 line 5
```

Below you will see other, more complicated examples. However, if you can solve the problem only for the examples in which the heights of the cards are equal (such as the one above), you can easily get 90 points out of 100. See the Grading for details.

Example Run 2 (Case 2)

```
This program will draw two cards next to each other.
──────────────────────
Texts must not be empty.
Text of first card: YOU CAN DO IT!
Text of second card: :)
──────────────────────
Width of first card must be at least 16.
Width of first card: 20
Width of second card must be at least 4.
Width of second card: 6
──────────────────────
Heights must be odd and at least 3.
Height of first card: 5
Height of second card: 3
──────────────────────
┌──────────────────┐┌────┐
│  YOU CAN DO IT!  ││ :) │
└──────────────────┘└────┘
```

Let's annotate the lines, replace spaces around texts with '**x**'s, and replace border characters with '**+**'s to make it easier to understand:

```
++++++++++++++++++++       → Card 1 line 1
+xxxxxxxxxxxxxxxxxx++++++  → Card 1 line 2, card 2 line 1
+xxYOUxCANxDOxIT!xx+x:)x+  → Card 1 line 3, card 2 line 2
+xxxxxxxxxxxxxxxxxx++++++  → Card 1 line 4, card 2 line 3
++++++++++++++++++++       → Card 1 line 5
```

Example Run 3 (Case 3)

```
This program will draw two cards next to each other.
──────────────────────
Texts must not be empty.
Text of first card: Easier than
Text of second card: it looks
──────────────────────
Width of first card must be at least 13.
Width of first card: 14
Width of second card must be at least 10.
Width of second card: 11
──────────────────────
Heights must be odd and at least 3.
Height of first card: 5
Height of second card: 7
──────────────────────
┌────────────┐┌──────────┐
│Easier than ││it looks  │
└────────────┘│          │
              │          │
              │          │
              │          │
              └──────────┘
```

Let's annotate the lines, replace spaces around texts with '**x**'s, and replace border characters with '**+**'s to make it easier to understand:

```
xxxxxxxxxxxxx++++++++++  →                Card 2 line 1
++++++++++++++xxxxxxxxx+  → Card 1 line 1, card 2 line 2
+xxxxxxxxxxxx+xxxxxxxxx+  → Card 1 line 2, card 2 line 3
+Easierxthanx+itxlooksx+  → Card 1 line 3, card 2 line 4
+xxxxxxxxxxxx+xxxxxxxxx+  → Card 1 line 4, card 2 line 5
++++++++++++++xxxxxxxxx+  → Card 1 line 5, card 2 line 6
xxxxxxxxxxxxx++++++++++  →                Card 2 line 7
```

Example Run 4

```
This program will draw two cards next to each other.
──────────────────────
Texts must not be empty.
Text of first card: A
Text of second card: B
──────────────────────
Width of first card must be at least 3.
Width of first card: 3
Width of second card must be at least 3.
Width of second card: 3
──────────────────────
Heights must be odd and at least 3.
Height of first card: 3
Height of second card: 4
──────────────────────
ERROR: Invalid inputs.
```

Other Examples

If the problem is not clear and you are not sure how the output must look like for a specific set of inputs, then you can ask this in the Student Forum. (I will share the desired output for the requested inputs.)

Notes

- **Base code:** The base code shared with you contains a template for the solution as well as the code for getting user inputs.
- **"Box-drawing characters":** The character palette is provided in the given code.
- **Horizontal alignment:** Whenever there is a need for an extra space character, show it on the right-hand side of the text. Because, for example, you cannot have 1.5 spaces on each side. Instead, you should have 1 space on the left and 2 spaces on the right. Card 2 of "Example Run 1" illustrates this.

- **Total width:** Pay attention to the example runs: A vertical line lies in the intersection of two cards. This line is common. Thus, the actual width for the whole thing is (card1_width + card2_width - 1).
- **Assumptions (Good news):** As shown in the "Example Run 4" and given code, you are not responsible to draw the cards for invalid inputs. For example, when the heights are odd numbers, it is more difficult to draw them. Fortunately, the user is asked to enter odd heights.

Submission Rules

- You are given a code template. Change the parts marked with "INSERT YOUR CODE HERE" only. Do not edit the other parts of the code!
- Rename the Python file from ceng113_hw1.py to ceng113_hw1_studentno.py (e.g. ceng113_hw1_123456789.py). Note that it is all lowercase.
- This is your personal homework. Thus, teamwork is not allowed. If you have any questions, ask them the assistants directly. It would be best if you ask your questions during the weekdays as it may not be possible to get answers during the weekend.
- Do not add extra features. Anything extra will make your solution worse, not better. If this homework is not challenging enough for you can do this: When the text is too long that it does not fit in a line, split it into multiple lines and center all the lines. Additionally, you can colorize the border and the text. You can Google this. However, do that separately, and do not submit that code! That is not your homework.
- Submit your solution by the due date.

Grading

Please see the code. The following is how each "INSERT YOUR CODE HERE" is graded:

- **5 points (File name):** Rename the given file properly.
- **10 points (The minimum widths):** "Assign proper values to card1_min_width and card2_min_width."
- **5 points (Input validation):** "Check if there is a problem with the inputs."
- **70 points (Case 1):** The case with card1_height == card2_height. This is the simplest case and is very similar to a previous exercise. See the Appendix to remember this exercise.
- **5 points (Case 2):** The case with card1_height > card2_height.
- **5 points (Case 3):** The case with card1_height < card2_height.

Outputs for each case must be **exactly the same** as in the example runs including spaces and empty lines.

Getting the first 90 points should be relatively easy. However, getting 100 full points requires approximately triple the effort. My personal advice: Do not continue with case 2 or case 3 until you are sure that you solved case 1 correctly. You will get a worse grade if you do more things worse rather than fewer things better.

Appendix

Once you fully understand this example, the homework becomes much easier.

```
text = input("Text: ")
width = int(input("Width: "))
height = int(input("Height: "))
border = input("Border character: ")

"""
*********  --> top_line
*       *  \
*       *   |--> upper_lines
*       *  /
*   A   *  --> center_line
*       * \
*       *  |--> lower_lines (same as upper_lines if height is odd)
*       * /
********* --> bottom_line (same as top_line)
"""

vertical = height - 3  # Total number of lines in upper_lines and lower_lines
upper = vertical // 2  # Number of lines in upper_lines
lower = vertical - upper  # Number of lines in lower_lines

horizontal = width - 2 - len(text)  # Total number of spaces in center_line
left = horizontal // 2  # Number of spaces on the left of text, in center_line
right = horizontal - left  # Number of spaces on the right of text, in
center_line

top_line = border * width + "\n"
upper_lines = (border + " " * (width - 2) + border + "\n") * upper
center_line = border + " " * left + text + " " * right + border + "\n"
lower_lines = (border + " " * (width - 2) + border + "\n") * lower
bottom_line = border * width + "\n"
print(top_line + upper_lines + center_line + lower_lines + bottom_line)
```

```
Text: Hello!
Width: 50
Height: 7
Border character: *
**************************************************
*                                                *
*                                                *
*                     Hello!                     *
*                                                *
*                                                *
**************************************************
```