Card Folding Explanation

The problem given to us for this assignment was to take a card, subdivided into squares, up to a maximum of 10^5 squares, that can be either hole punched out, or left solid. The card is the folded along a column so that the squares overlap, and wherever a hole overlaps another hole, a new hole can be seen (a hole is blocked by a no hole square and the un-punched squares will block each other). The time limit was given to be under 2 seconds.

The strategy of the designed algorithm was to first take the overhang (the part of the card that did not overlap with the other half when folded) and append it to the start of the array and the reverse it, depending on which side of the card the overhang is. Next, we deal with the section that overlaps when folded, we increment up the string from the start point while we simultaneously decrement down the string from the end point and compare values, if the value is the same then append it to the result, if not append 'x'.

Base on this logic we first have to determine if the fold point was before or past the halfway point of the card. Depending on if it was before or after we run a slightly different algorithm. If the overhang exists on the right side of the card, we reverse the overhang that we append to the result, else we do not.

Logic:

4 7 3

oxoooxo

xoxxoox

xxxoxxo

xoooxox

Folded right after the third column

Overhang: o x o x	Overlap: oxooox xoxxoo xxxoxx xoooxo	Comparison for first line: oxooox oxooox oxooox result of line: xxo overhang + result: oxxo

Result:

oxxo

xxox

oxxx

xxxo

Pseudo code:

```
Read_file_params()
Read_in_array[rows] //array of strings that store the input card
Result[rows] =”” //stores the result after evaluation of the Read_in_array
If(Read_in_array[0].length >= foldPoint){
        For( i=0; <rows; i++)
                For( j=foldPoint *2; j< Read_in_array[0].length; j++)
                        Result[i] += Read_in_array [i][j]
                Result[i] = result[i].reverse()
        For( i=0; <rows; i++)
                For( j=0 *2; j<foldPoint; j++)
                        If(Read_in_array [i][j] == Read_in_array [i][foldPoint*2-j-1]
                                Result[i] += Read_in_array[i][j]
                        Else
                                Result[i]+= ‘x’

}else{
        Start = foldPoint*2- Read_in_array[0].length
        For( i=0; <rows; i++)
                For( j=0; j<Start; j++)
                        Result[i] += Read_in_array [i][j]
        For( i=0; <rows; i++)
                Y=0
                For( j=start *2; j< foldPoint; j++)
                        If(Read_in_array [i][j] == Read_in_array [i][ Read_in_array[0].length -Y-1]
                                Result[i] += Read_in_array[i][j]
                        Else
                                Result[i]+= ‘x’
                        Y++
}
```

! Disclaimer!: Read code for full comments

The execution of the python implementation of this on my computer was approx 0.02sec which is under the 2 second requirement.