Section 18 **PDS Lab Test 2** **Oct 29, 2019**
Even set **Full marks: $100 \times 4 = 400$** **Time: 3 hours**

Note:

1. Marks will be deducted for inefficient coding, bad structuring of code, bad indentation, lack of important commenting, and deviation from input and output formats as shown in the examples.
2. You can use any library function unless forbidden in the question.
3. Name and submit your files as `e2a.c, e2b.c, e2c.c, e2d.c`.
4. In each file you must write your name, roll number, and machine number in the beginning as comment lines.

e2a. **(Sum of consecutive even squares)** Given a positive integer n, check whether it can be expressed as the sum of squares of two consecutive even integers.
You **cannot** use any loop.

Examples:

```
Enter n: 20       Enter n: 34         Enter n: 23113      Enter n: 23764
2 4               Does not exist.     Does not exist.     108 110
```

e2b. **(Number of stanzas)** Given a poem in a text file containing "`---End---`" at the end, count the number of stanzas in the poem.

Examples:

```
./a.out < poem1.txt          ./a.out < poem2.txt          ./a.out < poem3.txt
#stanzas = 2                 #stanzas = 4                 #stanzas = 4
```

e2c. **(Number count)** Given a positive integer $n(> 1)$, count the numbers that can be formed with the digits 2 and 3 such that the sum of digits is n — do it both recursively and iteratively.
For example, for $n = 8$, the numbers are $2222, 233, 323, 332$, and so the count is 4. (50+50)

Examples:

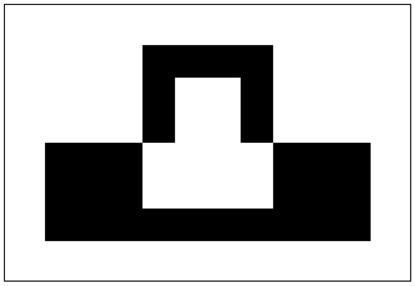
```
Enter n (>1): 5              Enter n (>1): 8              Enter n (>1): 11
count (recursive) = 2        count (recursive) = 4        count (recursive) = 9
count (iterative) = 2        count (iterative) = 4        count (iterative) = 9
```

e2d. **(Overlapping rectangles)** An input file (`rect1.txt` or `rect2.txt`) contains the information about n axis-parallel rectangles. Its 1st line contains the value of n, and each of the next n lines contains the (x, y) coordinates (all are integers) of two diagonally opposite corners (bottom-left and top-right) of a rectangle. Your program has to print on the terminal the areas of all these rectangles.

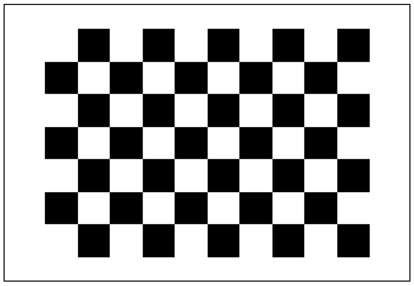
Now prepare an output binary image file named `e2d.pbm`. Image height = 400, width = 600, and each point of the image is black if and only if it belongs to exactly one rectangle.

You **cannot** use math library. (40 + 60)

Examples:



```
./a.out < rect1.txt
Areas = 75000, 50000, 15000.
```



```
./a.out < rect2.txt
Areas = 25000, 25000, 25000, 25000, 17500,
        17500, 17500, 17500, 17500.
```

Note: A `pbm` file contains “`P1`” in the 1st line, #columns and #rows in the 2nd line, and then each line represents a row of the image with ‘1’ = black and ‘0’ = white.