Fundamentals of Programming of {{ETSII, ETSIS, ETSIC, Double D.}}

Final Exam, Feb 11th 2022

Surname, Name _____ ComputerNumber _____ Degree _____

After Uploading the source files, at the end, hand this sheet in filled with your Surname, Name, Computer Number, and the Name of your degree

Filenames:

`1.pascal.cpp` `2.inorder.cpp` `3.img.cpp`

At the end of the exam you will only need to upload these 3 `.cpp` files to the corresponding task on the Campus Virtual.

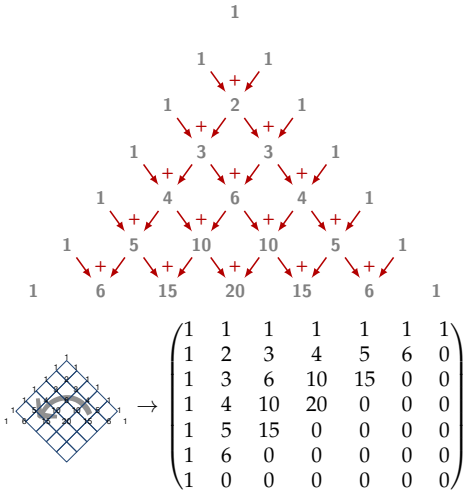
If on Terminal, remember: Start on Terminal, then:

1. `cd Desktop`
2. `subl 1.pascal.cpp` (then...Save: command-S ⌘-S)
3. `c++ 1.pascal.cpp`
4. `./a.out`

Do not forget to write your `Surname, Name; Degree; Group; Machine Number` at the top of every `.cpp` file, each thing in a different line

Do not use any external disks, documentation, or resource except your brains. Except a lecturer, do not speak with anyone else

1 **2.5pts** `1.pascal.cpp` Pascal's triangle is an infinite symmetric triangle of integer numbers. Its first row has only one figure, the number 1. The second and all the rest of the rows start in 1 and end in 1, but the numbers in the middle are computed summing the two numbers over each of them. Its first `N`=7 rows then are:



$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 0 \\ 1 & 3 & 6 & 10 & 15 & 0 & 0 \\ 1 & 4 & 10 & 20 & 0 & 0 & 0 \\ 1 & 5 & 15 & 0 & 0 & 0 & 0 \\ 1 & 6 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Rotating 45º anticlockwise the triangle, we get a `NxN` matrix for which the upper-left triangle is the pascal's and the the lower-right triangle can be filled with 0s. In this way, the first row and the first column are 1s; the numbers at the right of the anti-diagonal are 0s; and the rest of the numbers are computed as the sum of the number above it and at its left.

Design an algorithm to fill this matrix (a `TSqMat`, size `N`) and another to show that matrix on the screen. For `N`=7 the output should be:

```
1 1 1 1 1 1 1
1 2 3 4 5 6
1 3 6 10 15
1 4 10 20
1 5 15
1 6
1
```

2 **3.5pts** `2.inorder.cpp` Design a program that *asks the user for a series of words* until the word `"end"` is entered. Saving the non-repeated words that are *'in order'*, that is the words for which their letters are all in order ('a'...'z') inside the word (there can be repeated letters in the words). Example:

```
[INPUT]
Enter a series. end finishes the input:
i know how all of it is no half of it is not to do nor aim any end

[OUTPUT]
Unique in-sorted words:
i know how all it is no not do nor aim any
```

For this program suppose: (1) There can be an infinite number of words on the input; (2) a final `end` word will be entered; (3) all the words are lowercase (and unlimited in the number of chars); (4) there will be no more than `MAXNUMSORTEDDIFFWORDS` `inOrder` different words; (5) in between words there will be only spaces.

3 **4pts** `3.img.cpp` We want to scale-up images. An image is a matrix `TMat m` of `NRows x NCols` integer numbers in which a number `>= 1` represents a color. To scale-up the image a new `TMat2 m2` with double number of rows: `(2 NRows) x NCols`. To fill `m2` follow the next steps:

1. compute the **mean** (as integer, `mMean`) of all the elements of `m`
2. the rows of `m` are copied to the **even rows** of `m2`: row 0 to row 0, row 1 to row 2, row 2 to row 4...
3. **odd rows** of `m2` are initialised to 0 and then,
4. take the mean of surrounding **neighbours** of each cell in these odd rows. If a neighbour still has the value 0 (remember the odd rows started with 0), use `mMean` instead that 0. If a cell is inside the matrix take its value (or `mMean` if 0) and add 1 to the count. You will count 3 neighbour cells for corners, 5 for no-corner-side-cells and 8 for inner cells.

Example:

$$m = \begin{pmatrix} 3 & 2 & 5 \\ 6 & 5 & 3 \end{pmatrix} \Rightarrow m2 = \begin{pmatrix} 3 & 2 & 5 \\ 0 & 0 & 0 \\ 6 & 5 & 3 \\ 0 & 0 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 3 & 2 & 5 \\ 4 & 4 & 3 \\ 6 & 5 & 3 \\ 5 & 4 & 4 \end{pmatrix}$$

We have that `mMean` is $4 = (3+2+5+6+5+3)/6$. For the cell `m2[1][0]`, the mean of its 5 neighbours (3, 2, 0, 5, 6), after substituting the 0 by 4, $(3+2+4+5+6)/5$ is 4. For the cell `m2[1,1]` we have 8 neighbours: (3, 2, 5, 0, 0, 6, 5, 3), after the 0-substitution: $(3+2+5+4+4+6+5+3)/8 = 4$ for the cell `m2[1][2]` we have 5 neighbours $((2+5+4+5+3)/5 = 3)$. This process must also be taken for the last row.

Design a function `scaleup()` that receives an image `TMat` and returns a scaled `TMat2`.

In the `main()` 'quick-fill' a `TMat` with the values of the example and then `print` the original and `print2` the resulting matrix. *Do not ask the user for numbers.*

HINT: Organise the code in subprograms for each subtask. Be careful to not read/write out of the matrix boundaries.