be-OI 2026 **Final - SENIOR** Saturday, March 14, 2026	**Fill in this box in CAPITAL LETTERS please** FIRST NAME : .... LAST NAME : .... SCHOOL :	O **Reserved**

Finals of the Belgian Olympiad in Informatics 2026 (duration: **2h**)

Instructions to read carefully before the exam.

1. **Wait for the starting signal** before removing the sheets from the plastic sleeve.
2. You must take the finals in the same language as the one in which you took the qualification round. **Check that this page is in the correct language.** If not, report it or change seats.
3. Check that you have received the correct **set of questions** mentioned above in the header.
   - For students up to and including the second year of secondary school: category **cadet**, **pink** sheets.
   - For students in the third or fourth year of secondary school: category **junior**, **green** sheets.
   - For students in the fifth year of secondary school and above: category **senior**, **yellow** sheets.
4. When the starting signal is given, take the sheets out of the sleeve, but **do not remove the staples.**
5. Write your last name, first name and school **very legibly in CAPITALS and only on this page**.
6. Write your **answers on the coloured sheets** stapled with this page.
   Write **clearly and legibly** using a blue or black **pen or bic**.
7. Use the white sheets as scratch paper.
   Do not forget to copy your solutions onto the coloured sheets.
8. The exam lasts 2 hours and you must remain seated until the end.
   If you finish early, call an invigilator to hand in your answers.
9. When the finishing signal is given you must immediately hand in your solutions.
   - Do not put the sheets back in the plastic sleeve.
   - **Place the coloured sheets, still stapled**, in one of the boxes provided for that purpose.
   - **Return the plastic sleeve** by placing it in a box provided for that purpose.
   - Keep the white sheets.
10. You may only have writing materials with you. Calculators, smartphones, ... are **forbidden**.
11. All the code snippets in the problem statements are in **pseudo-code**. On the following pages, you will find a **description** of the pseudo-code that we use. If you have to respond with code, you can use **pseudo-code** or a **common programming language** (Java, C, C++, Pascal, Python, ...). Syntax errors are not taken into account for grading.

The Belgian Olympiad in Informatics is possible thanks to the support of our members:

Pseudo-code cheat sheet

Data is stored in variables, each identified by a name.
The name is used to access the variable to store or retrieve data.
We use ← as the assignment operator to store a value in a variable.
Example: `n←100` places the integer `100` in the variable named `n`.

Variables and simple data	Examples
integer	`n←100` `m←-1`
real number	`pi←3.1415` `z←0.0`
logical value (boolean)	`t←`**`true`** `t←`**`false`**

Arithmetic operations can be performed using numbers and variables.

addition and subtraction with + and − (example: `a+3`)
multiplication with ⋆ or × (example: `2*a*b`)
integer division: quotient with / or // and remainder with % (example: `14//3` is equal to `4` and `14%3` is equal to `2`)
non-integer division /: rarely used, indicated when applicable (example: `3.6/4.0` is equal to `0.9`)
power or exponent with ^ (example: `x^3` is equal to `x*x*x`)

Variables are often used to compute a result
and then store it in a variable.
Sometimes the result is stored in one of the variables used in the computation.
The box on the right contains code that illustrates this.
After executing this code, `x=25`, `a=4` and `b=10`.

```
a ← 3
b ← 5
x ← a*b + 10
a ← a + 1
b ← b*2
```

```
if (a < b)
    { p ← p+5 }
else
    { p ← p-2 }
c ← c-1
```

The **`if`** instruction allows executing code only if a condition is true.
Optionally, the **`else`** instruction can be added to execute different code only if the condition is false.
In the example on the left, if the number in variable `a` is less than the one in variable `b`, then `5` is added to variable `p`,
otherwise `2` is subtracted from variable `p`.
Then, in all cases, `1` is subtracted from variable `c`.

The instructions to be executed in the different cases will be clearly identified
either by placing them between braces { }, or by indentation.
Most often, both will be used: braces and indentation, as above.

Here is the list of the most common comparison operators.

= or ==	<	<= or ≤	>	>= or ≥	!= or ≠
is equal to	less than	less than or equal	greater than	greater than or equal	is not equal to

Multiple conditions can be tested by combining them with the logical operators **and**, **or**, **not**.

The condition `(p and q)` is true if both conditions `p` and `q` are true.
The condition `(p or q)` is true if at least one of the 2 conditions is true.
The condition `not(p)` is true if `p` is false and false if `p` is true.
The logical value of a condition can be stored in a boolean variable for later use. Example: `f←((a==5) and (b>=0)) or ((a<0) and (b!=10))`
The condition of an **if** is sometimes a simple boolean variable. Example: `if (f) {a←10} else {b←10}`

Sometimes it is necessary to store multiple data items in a single structured variable.

Structured variables and data	Examples
array, list, vector	`seq←[7,11,0,-4,9]`
array, matrix	`M←[[0,1,2],[3,-1,3],[0,0,5]]`
pair/tuple	`coord←(1,7)`
text	`n←"John"`

Individual elements of a structured variable are identified by an index written in square brackets after the variable name.
The first element of a structured variable `G` has index `0` and is denoted `G[0]`.
The second element has index `1` and the last has index `n-1` if there are `n` elements in total.
The number of elements in a structured variable is returned by the `len()` function.
Example: if `G=[5,9,12]` then `len(G)=3`, `G[0]=5`, `G[1]=9` and `G[2]=12`.
The array has size `3`, but the highest index is `2`.

To repeat code, for example to iterate over the elements of an array, a **for** loop can be used.

```
sum ← 0
n ← len(T)
for (i ← 0 to n - 1 step 1)
{
    sum ← sum + T[i]
}
```

The notation `for (i←a to b step k)` represents a loop

- in which `i` starts at the value `a`,
- that is repeated as long as `i≤b`,
- that increases `i` by `k` at the end of each step.

The example on the left computes the sum of the elements of the array `T`. The elements are added one by one into the variable `sum`.

The instructions to be executed in a loop will be clearly identified either by placing them between braces { }, or by indentation. Most often, both will be used: braces and indentation, as above.

A loop can also be written using the **while** instruction, which repeats code as long as its condition is true.

In the example on the right, a positive integer `n` is divided by `2`,
then by `3`, then by `4` ...
until it consists of only a single digit
(i.e. until `n<10`).

```
d ← 2
while (n ≥ 10) {
    n ← n/d
    d ← d+1
}
```

Author: *Ludovic Galant*

Question 1 – The Lost Palace episode 1: counting.

Archaeologists have discovered an ancient palace with many rooms.
But there are traps that they cannot avoid.
They have called Professor Zarbi to the rescue.

The professor quickly understood that one must walk on certain engraved tiles to deactivate the traps.
In the rooms, the engraved tiles are always arranged in several horizontal rows.
The first row consists of a single engraved tile, called the *top*.

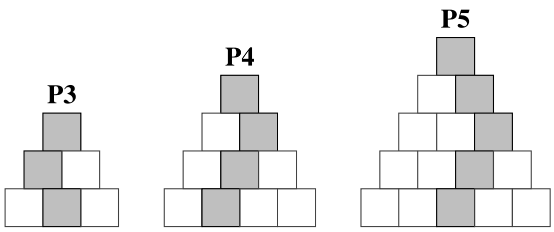
The professor discovered that one must advance while following certain rules, otherwise a trap is inevitably triggered.

1. You must start from the top (at the top in the examples) and walk only on engraved tiles.
2. When advancing, you must always move from one engraved tile to another that touches it (not just by a corner).
3. You must pass through exactly one engraved tile in each row.
4. You must reach the last row of engraved tiles (at the bottom in the examples).

From now on, the word ***path*** means a traversal that follows these 4 rules.
From now on, all the ***tiles*** mentioned or drawn are engraved tiles.

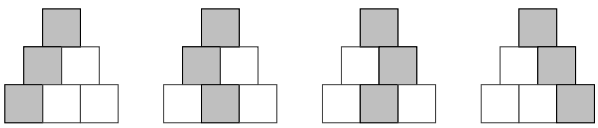
The tiles are often arranged in a pyramid, as in all the examples on this page.
We denote **Pn** a pyramid consisting of n rows of tiles.

Example 1
Here are pyramids **P3**, **P4** and **P5**.
An example of a path is coloured in grey in each one.
The top is grey (rule 1).
The grey tiles of adjacent rows touch each other (rule 2).
There is exactly one grey tile per row (rule 3).
There is a grey tile in the last row (rule 4).



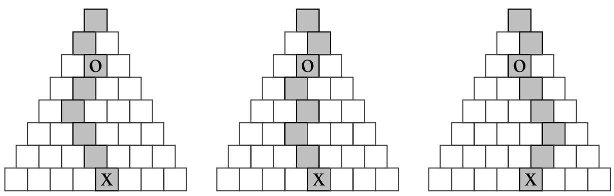
Professor Zarbi wants to count how many different paths allow one to traverse pyramids.

Example 2
Here are all the paths that traverse **P3**.
1 path reaches the left tile of the last row.
2 paths reach the centre tile of the last row.
1 path reaches the right tile of the last row.
In total, 4 different paths traverse **P3**.



Sometimes, he wants to count only the paths that pass through certain tiles.

Example 3
Here are 3 paths that traverse **P8**.
They all pass through the second tile of the third row
(tile marked with an o).
They all reach the fifth tile of the last row
(tile marked with an x).



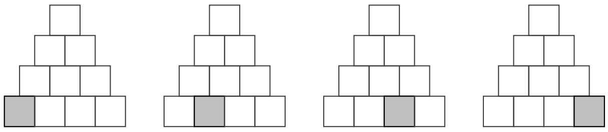
Note: on one of the last pages of this booklet,
blank pyramids are available for your scratch work.

Counting the paths that reach a specific tile.

In the following questions, pyramids are drawn with 1 grey tile in the last row.
How many different paths reach the grey tile? Write the answers in the rectangles.

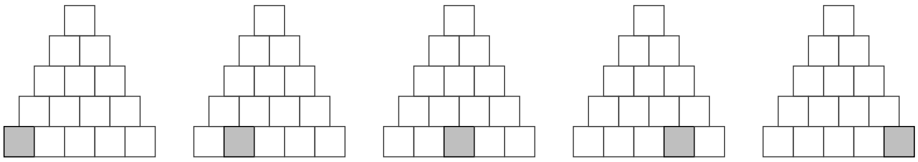
Q1(a) /4	**How many paths reach the grey tile in P4?**

1	3	3	1



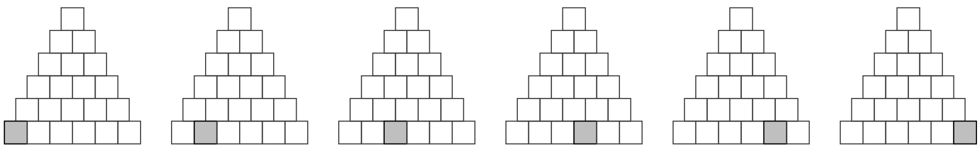
Q1(b) /5	**How many paths reach the grey tile in P5?**

1	4	6	4	1



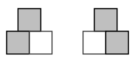
Q1(c) /6	**How many paths reach the grey tile in P6?**

1	5	10	10	5	1

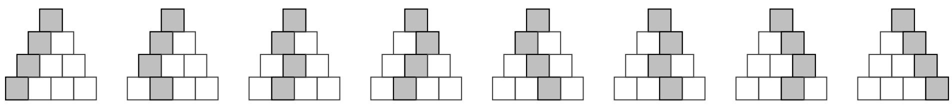


Counting all the paths.

Q1(d) /1	**In total, how many paths traverse P2? Solution : 2**



Q1(e) /1	**In total, how many paths traverse P4? Solution : 8**



Q1(f) /1	**In total, how many paths traverse P5? Solution : 16**

Q1(g) /1	**In total, how many paths traverse P6? Solution : 32**

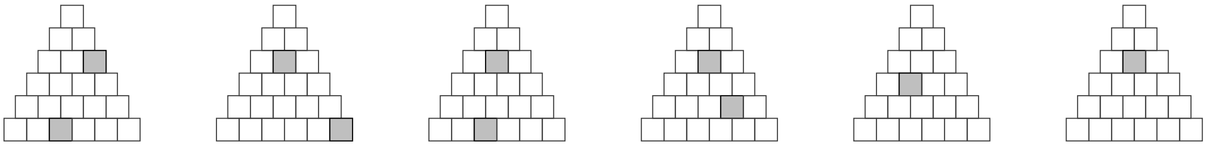
Q1(h) /1	**In total, how many paths traverse Pn (where n is a positive integer)? Solution :** 2^{n-1}

Counting the paths that pass through certain tiles.

In the following questions, pyramids are drawn with one or more grey tiles.
Write in the rectangle above the pyramid the number of different paths that pass through the grey tiles. **Don't forget the 4 rules!**

Q1(i) /6	**How many paths pass through the grey tiles of the P6 pyramids?**

1*1=1	2*0=0	2*3=6	2*1*2=4	3*4=12	2*8=16

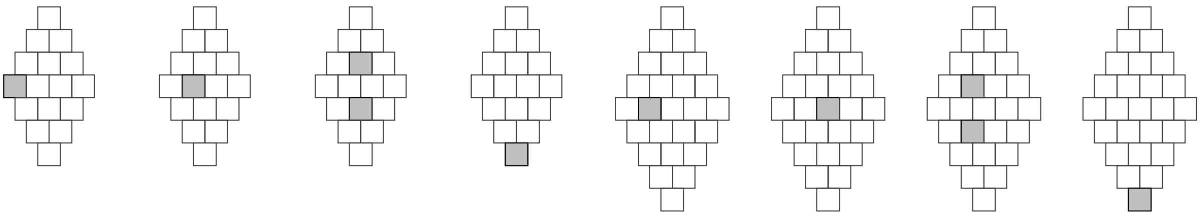


Tiles arranged in a diamond.

The tiles are often arranged in a pyramid, but this is not always the case.
There is for example a diamond arrangement where the last row has only one tile.
In the following questions, diamond rooms are drawn with one or more grey tiles.
Write in the rectangle above the diamond the number of different paths that pass through the grey tiles.

Q1(j) /8	**How many paths pass through the grey tiles of the diamonds?**

1	9	8	20	16	36	18	70

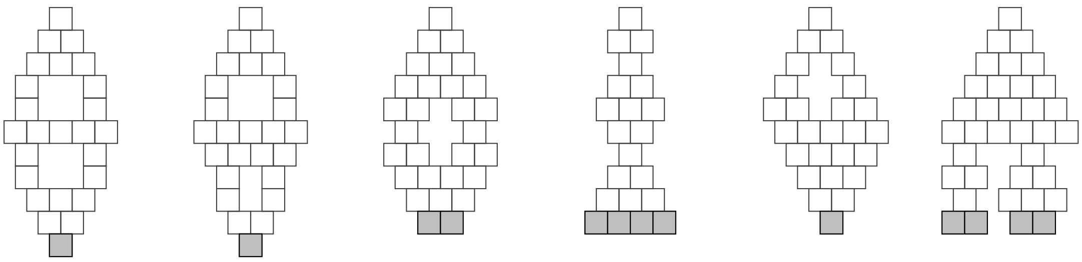


Strange rooms.

The tiles of certain rooms are arranged in a complicated way.
In the following questions, you must count all the paths that traverse strange rooms.
Write in the rectangle above the room the total number of paths that traverse it (up to a grey tile).

Q1(k) /6	**In total, how many paths traverse these strange rooms?**

4	6	50	96	21	114



Steps in large pyramids.

Some rooms contain very large pyramids of tiles and Professor Zarbi wants to find a way to simplify the path counting. He uses the conventions below.

- The `n` rows of tiles of a pyramid **Pn** are numbered from `0` to `n−1` starting from the top.
- In a pyramid, row number `r` always has `r+1` tiles numbered from `0` to `r` starting from the left.
- The coordinates of a tile are `(r,c)` where `r` is its row and `c` is its number within its row.

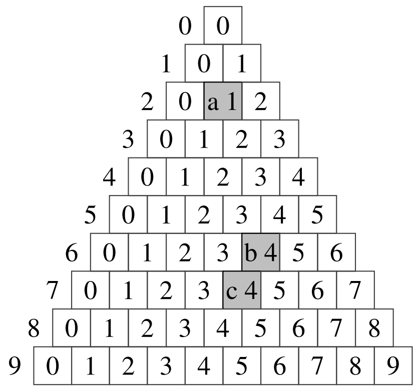
Example

The figure opposite represents a pyramid **P10**.
The row numbers are displayed on the left.
The numbers used in each row are displayed on the tiles.
The 3 grey tiles are identified with the letters a, b and c.
Their coordinates are `(2,1)`, `(6,4)` and `(7,4)`.

Note

This figure also gives the coordinates
of the tiles in the smaller pyramids.
For example, the first 4 rows show
the coordinates in a pyramid **P4**.

P10



Professor Zarbi counts in several steps the paths passing through grey tiles of a large pyramid.
Each step uses a sub-pyramid.

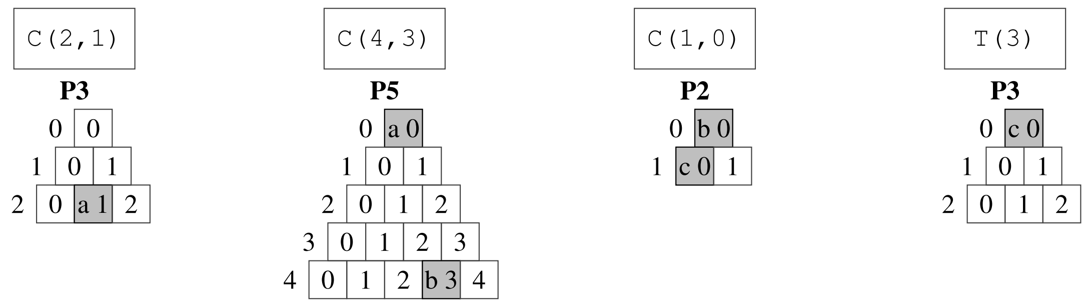
- The first sub-pyramid has the same top as the large pyramid,
  and its last row contains the first grey tile of the large pyramid.
- Then, sub-pyramids are used whose top is a grey tile,
  and whose last row contains the next grey tile.
- There is a special case if the last row of the large pyramid does not contain a grey tile.
  In that case, there is no grey tile in the last row of the last sub-pyramid.

Professor Zarbi uses the following notations.

- `C(r,c)` is the number of paths reaching tile `(r,c)` (in a pyramid with `r+1` rows).
- `T(n)` is the total number of paths that traverse a pyramid **Pn** (i.e. with `n` rows).
  Warning: for the purposes of this **BeOI** final, it is **forbidden to use `T(1)`**.

Example

Here is the decomposition into 4 steps of the path counting in **P10** above.
First step: from the top to the first grey tile a.
Second step: from the first grey tile a to the second b.
Third step: from the second grey tile b to the third c.
Last step: from the last grey tile c to the last row of **P10**.
The number of possible paths for each step is written in the rectangle above the step.

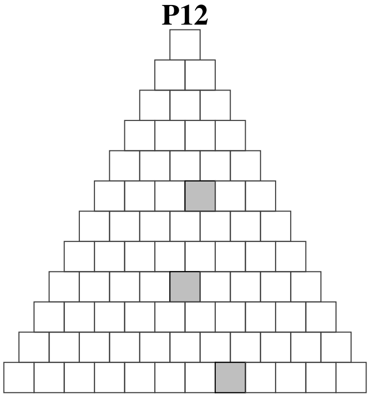


Obviously, one must use the numbers of possible paths from the steps to determine the number of paths passing through the grey tiles in the large pyramid.
In the following questions, give a mathematical expression using the notations `C(r,c)` and `T(n)` to determine the number of paths that pass through the grey tiles of the **P10** example from the previous page and of the pyramids **P12**, **P13** and **P15** below.

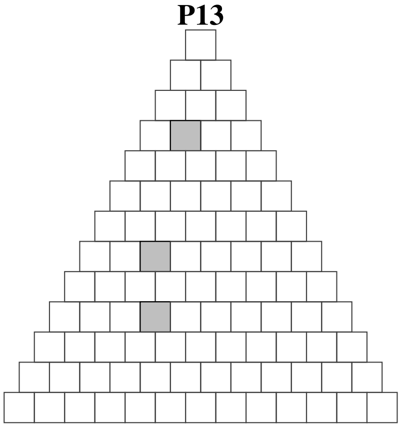
Important note
Do not compute the final answer !
You must write a mathematical expression resembling for example `C(5,2)*T(3)+C(4,2)`.

Q1(l) /2	**Expression for the number of paths passing through the grey tiles of P10 from the example.** **Solution :** `C(2,1) * C(4,3) * C(1,0) * T(3)`
Q1(m) /2	**Expression for the number of paths passing through the grey tiles of P12.** **Solution :** `C(5,3) * C(3,1) * C(3,3)`
Q1(n) /2	**Expression for the number of paths passing through the grey tiles of P13.** **Solution :** `C(3,1) * C(4,1) * C(2,1) * T(4)`
Q1(o) /2	**Expression for the number of paths passing through the grey tiles of P15.** **Solution :** `C(3,2) * C(4,2) * C(3,0) * C(3,1) * T(2)`

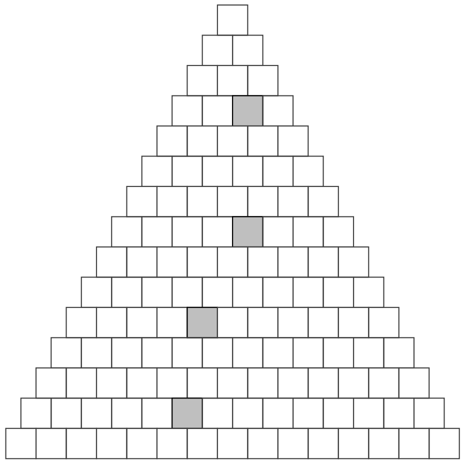
P12



P13



P15



Automation.

Professor Zarbi has written a function `countPyra(n,G)` to automate the counting of the number of paths passing through grey tiles in a large pyramid.

This function uses 2 properly initialised parameters.

- `n`: an integer that contains the number of rows of the large pyramid.
- `G`: an array of pairs `(r,c)` that contains the coordinates of the grey tiles (in order from top to bottom).

The program also uses the functions `C(r,c)` and `T(n)` which return the values explained previously.
Finally, the program uses a loop "**for** (r,c) **in** G" which executes as many times as there are elements in `G`.

Examples with the pyramid P10 drawn earlier in this document.

- `n=10`
- `G=[(2,1), (6,4), (7,4)]`
  So `G[0]=(2,1)`, `G[1]=(6,4)`, `G[2]=(7,4)`.
- `(r0,c0)←(0,0)` initialises the 2 variables at the same time (this is equivalent to `r0←0` and `c0←0`).
- `C(2,1)` returns `2` and `T(3)` returns `4`.
- The loop "**for** (r,c) **in** G" executes 3 times.
  The first time `(r,c)=(2,1)`, the second time `(r,c)=(6,4)`, the last time `(r,c)=(7,4)`.

Professor Zarbi writes a first version of the program that does not perform any verification.
This simple program works correctly with the 4 large pyramids **P10**, **P12**, **P13** and **P15** provided that `n` and `G` are properly initialised.

The program uses the following variables.

- `nbr`: the number of paths passing through the grey tiles of the large pyramid.
- `h`: the number of the last row of the sub-pyramid currently being processed (which therefore has `h+1` rows).
- `k`: the number of the grey tile in the last row of the current sub-pyramid (with $0 \le k \le h$).

The following question is worth between 0 and 6 points.
You lose 2 points for each blank ____ that is not correctly completed. There are no negative points.

Q1(p) /6	Complete the ____ in the function **countPyraV1(n,G)**.

```
function countPyraV1(n,G) {
  nbr ← 1
  (r0,c0) ← (0,0)

  for (r,c) in G {
    h ← r - r0

    k ← c - c0

    nbr ← nbr * C(h,k)

    (r0,c0) ← (r,c)
  }
  if (r0 < n-1) { nbr ← nbr * T(n-r0) }
  return nbr
}
```

Some pyramids can cause problems.

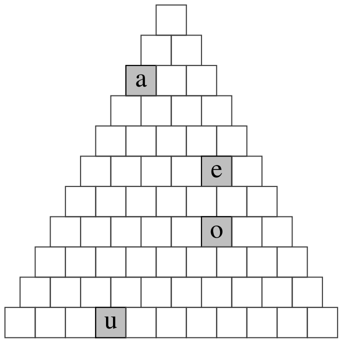
Example

There are no paths passing through all the grey tiles of the pyramid opposite.

It is impossible to go from a to e.

Similarly, there is no sub-pyramid with top o whose last row contains u.

P11



The program `countPyraV1` does not account for these cases and may give wrong answers or stop with an error message!
The program `countPyraV2` is an improvement that gives the correct answer in all cases.
Professor Zarbi has only replaced the line of code `nbr←` ______ from `countPyraV1` with a test in `countPyraV2`.
What are the conditions to check and what should be done depending on whether they are true or false?
It is up to you to write the test in the following question.

This question is worth between 0 and 6 points.
You lose 2 points for each blank ______ that is not correctly completed. There are no negative points.

Q1(q) /6	Complete the ______ in the test of **`countPyraV2`** that replaces a line of **`countPyraV1`**.

```
if (0 <= k and k <= h)
    {nbr ← nbr * C(h,k)}
else
    {nbr ← 0}
```

Question 2 – The Lost Palace episode 2: decrypting.

Professor Zarbi's discoveries do not yet make it possible to avoid all the traps.
But the archaeologists have discovered the room plans among other important documents.
In these plans, a pyramid marked with small discs is drawn on each tile.
The professor understands that these pyramids represent numbers and he manages to decrypt them.

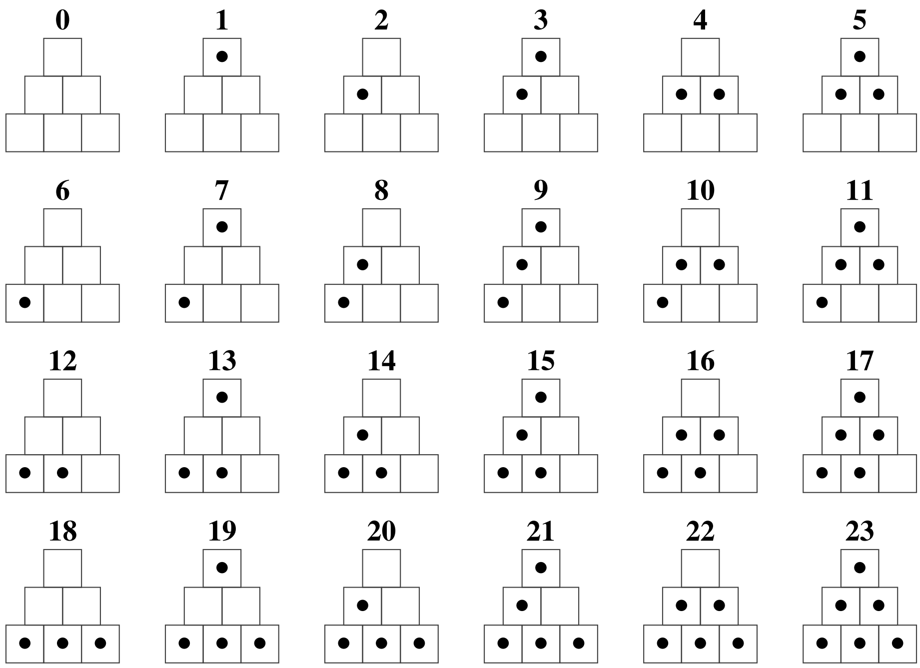
Warning! Do not confuse!
In episode 1, the pyramids were made of tiles.
In episode 2, the pyramids are numbers marked on the tiles.

The numbering system.

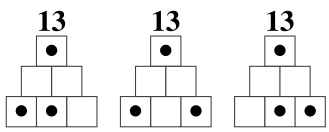
Numbers are represented by pyramids in which certain cells contain a small black disc.

- 0 is represented by a pyramid with no disc at all.
- To go from one pyramid to the next, that is from one number to the next, proceed as follows.
  1. Starting from the top, find the first row that contains an empty cell.
  2. Place a small disc in any empty cell of that row.
  3. Empty all the cells of the rows above (those that were filled).

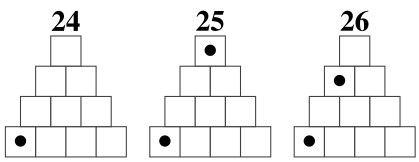
Examples: here are **P3** pyramids representing the numbers from 0 to 23.



Numbers that have multiple representations.
Several pyramids can represent the same number
since the black discs can be placed
anywhere in their row (point 2 above).
Example: the 3 pyramids shown here all represent the number 13.



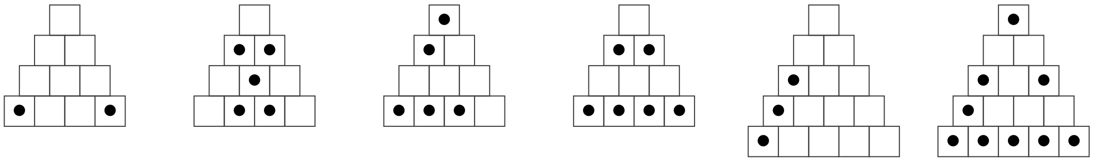
Representations of larger numbers.
23 is the largest number that can be represented by a **P3** pyramid.
To go further, one must use pyramids with more rows.



Note: ***drawing a pyramid*** means ***drawing small black discs*** **in certain cells of it**.
Special case: if no disc is drawn, one *draws an empty pyramid* which represents the number 0.

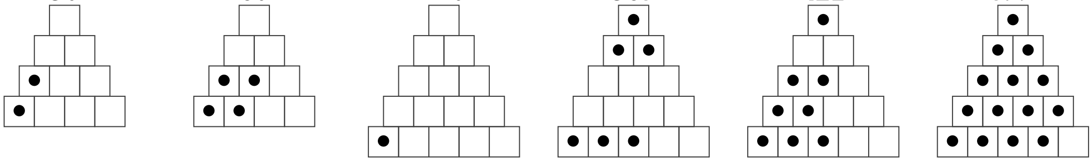
Q2(a) /6	**Write in each rectangle the number represented by the drawn pyramid.**

48 58 75 100 150 637



Q2(b) /6	**Draw a pyramid representing the given number.**

30 **60** **120** **365** **421** **599**



Above, it was noted that several pyramids can represent the same number and that 23 is the largest that can be represented by a **P3** pyramid. This inspires the following questions, which you can answer with a number or a mathematical expression.

Q2(c) /1	**How many different P3 pyramids can be drawn?** Solution : $2^6 = 64$

Q2(d) /1	**How many different P4 pyramids can be drawn?** Solution : $2^{10} = 1024$

Q2(e) /1	**What is the largest number that can be represented with a P4 pyramid?** Solution : $5! - 1 = 119$

Q2(f) /1	**How many different P5 pyramids can be drawn?** Solution : $2^{15} = 32768$

Q2(g) /1	**What is the largest number that can be represented with a P5 pyramid?** Solution : $6! - 1 = 719$

Q2(h) /1	**How many different P6 pyramids can be drawn?** Solution : $2^{21} = 2097152$

Q2(i) /1	**What is the largest number that can be represented with a P6 pyramid?** Solution : $7! - 1 = 5039$

In the following questions, you must give all the numbers that have a certain property.
If there are several numbers, write them separated by commas.
If there is no such number, answer by drawing a cross ×.

Q2(j) /1	**Which numbers have exactly one representation in P3?** Solution : 0, 1, 4, 5, 18, 19, 22, 23

Q2(k) /1	**Which numbers have exactly 2 representations in P3?** Solution : 2, 3, 20, 21

Q2(l) /1	**Which numbers have exactly 3 representations in P3?** Solution : 6, 7, 10, 11, 12, 13, 16, 17

Q2(m) /1	**Which numbers have exactly 4 representations in P3?** Solution : X

Q2(n) /1	**Which numbers have exactly 5 representations in P3?** Solution : X

Q2(o) /1	**Which numbers have exactly 6 representations in P3?** Solution : 8, 9, 14, 15

Q2(p) /1	**Which numbers have exactly 36 representations in P4?** Solution : 56, 57, 62, 63

Q2(q) /1	**What is the smallest number that has exactly 10 representations in P5?** Solution : 122

Q2(r) /1	**What is the largest number that has exactly 10 representations in P5?** Solution : 479

New notation: list of a pyramid.

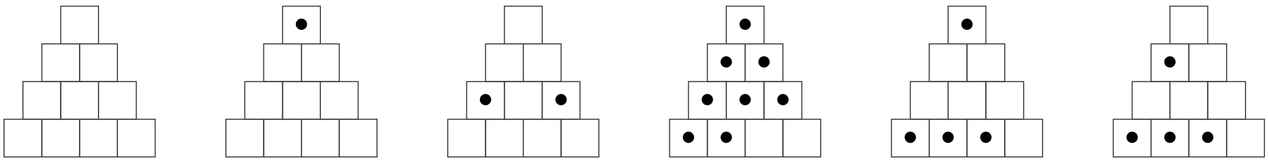
Drawing pyramids is tedious and impractical.
Instead of drawing a pyramid, Professor Zarbi prefers to give the list, starting from the top, of the numbers of black discs in each row.

From now on, the ***number of a pyramid*** is the number represented by the pyramid.
It is a positive integer, written as usual with the digits 0 to 9.

From now on, the ***list of a pyramid*** is the list of the numbers of black discs in each row of the pyramid.
This list starts with `0` if there is no black disc at the top or with `1` if there is one.
This list contains `n` numbers for a **Pn** pyramid.

Examples: here are some **P4** pyramids.
For each one, its **number** is shown above and its **list** below.

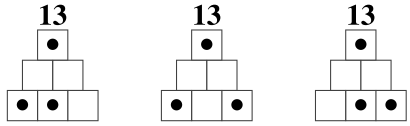
0 **1** **12** **71** **73** **74**



`[0,0,0,0]` `[1,0,0,0]` `[0,0,2,0]` `[1,2,3,2]` `[1,0,0,3]` `[0,1,0,3]`

Note.
If several pyramids represent the same number, then they have the same list.
Example: the 3 **P3** pyramids that represent the number 13 have the same list `[1,0,2]`.

13 **13** **13**



`[1,0,2]` `[1,0,2]` `[1,0,2]`

Automation.

The professor often needs to compute the number of a pyramid from its list.
He also sometimes needs, conversely, to compute the list of a pyramid from its number.
He wants to write programs to automate his calculations.

A few reminders.

- **Integer division.**
  The operator `/` (or `//` if you prefer) gives the integer quotient of a division between 2 integers.
  The operator `%` gives the remainder of a division between 2 integers.
  Example: `14/3` equals `4` (as does `14//3`) and `14%3` equals `2`.
- **Numbering of list elements.**
  The elements of a list are numbered starting from `0`.
  Example: if `LIS=[1,0,2]` then `LIS[0]=1`, `LIS[1]=0` and `LIS[2]=2`.
- **Initialising a list.**
  `LIS←[0]*n` creates a list of `n` zeros.

Function `PyrLIS(n,dec)`
This function takes 2 parameters.
- `n`: the number of rows of the pyramid.
- `dec`: a positive integer that must be represented by a pyramid.
  `dec` is named this way as a reminder that it is a **decimal** number (i.e. written with the 10 digits from 0 to 9).

The function returns the list of a **Pn** pyramid that represents the number `dec`.

Examples: `PyrLIS(3,13)` returns `[1,0,2]` and `PyrLIS(4,13)` returns `[1,0,2,0]`.

The following question is worth between 0 and 6 points.
You lose 3 points for each zone ______ that is not correctly completed.

Q2(s) /6	Complete the ______ in the function `PyrLIS(dec,n)`.

```
function PyrLIS(n,dec) {
  LIS ← [0]*n
  for r ← 0 to n-1 step 1 {
    LIS[r] ← dec % (r+2)

    dec ← dec // (r+2)
  }
  return(LIS)
}
```

Function `Pyrdec(n,LIS)`
This function takes 2 parameters.
- `n`: the number of rows of the pyramid.
- `LIS`: the list of a pyramid.

The function returns the number `dec` of the **Pn** pyramid whose list is given.

Examples: `Pyrdec(3,[1,0,2])` and `Pyrdec(4,[1,0,2,0])` return `13`.

The following question is worth between 0 and 6 points.
You lose 3 points for each zone ______ that is not correctly completed.

Q2(t) /6	Complete the ______ in the function `Pyrdec(LIS,n)`.

```
function Pyrdec(n,LIS) {
  dec ← 0
  v ← 1
  for r ← 0 to n-1 step 1 {
    v ← v * (r+1)

    dec ← dec + LIS[r] * v
  }
  return(dec)
}
```

Question 3 – The Lost Palace episode 3: adding.

Professor Zarbi has decrypted all the plans where small disc pyramids were drawn on the tiles.
He copied the plans by replacing each small pyramid with the number it represents.
After long deliberation and meticulous experiments, he has finally found the solution...

Warning! Do not confuse!

In episode 3, the pyramids are once again made of tiles, as in episode 1.
But in this episode, each tile bears a number (the one decrypted in episode 2).

Maximum sum paths.

From now on, a ***number pyramid*** is a pyramid with a number on each tile.

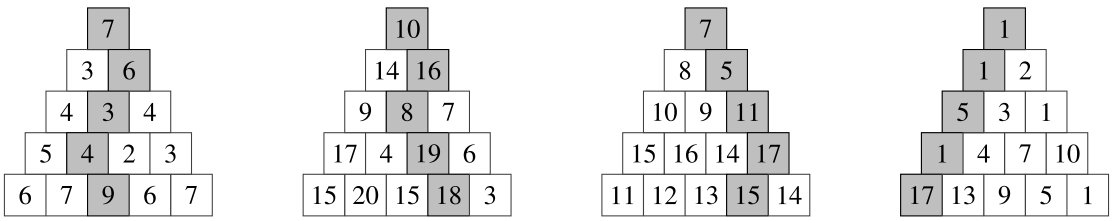
As a reminder, a ***path***

- starts from the top,
- moves from one tile to another that touches it,
- passes through exactly one tile in each row,
- must reach the last row.

The ***score*** of a path is the sum of the numbers marked on the tiles of that path.
Professor Zarbi discovered that the path with the highest score does not trigger a trap.
If several paths have the same maximum score, none of them triggers a trap.
Any path with a score lower than the maximum score triggers a trap.

Examples: here are some **P5** pyramids.
For each one, a path with maximum score is shaded in grey.
The score of the path is shown in the rectangle.

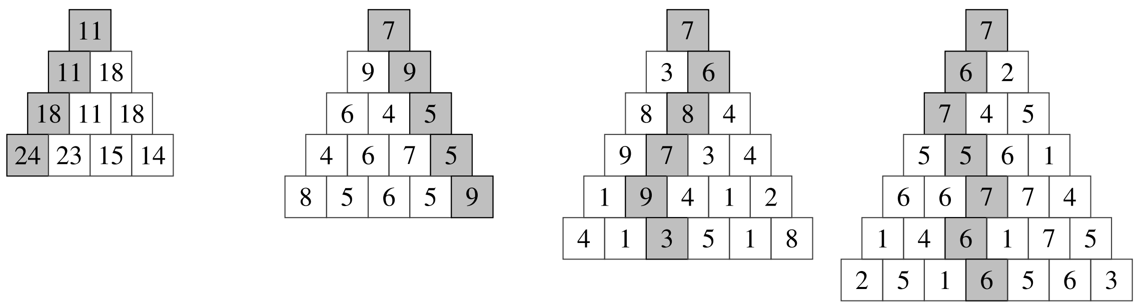
`7+6+3+4+9=29` `10+16+8+19+18=71` `7+5+11+17+15=55` `1+1+5+1+17=25`



In the following question, pyramids are drawn with a number on each tile.
In each one, trace a path with maximum score (by circling the numbers or shading the tiles with a pencil).
Write the score of the traced path in the rectangle above the pyramid.

Q3(a) /8	Trace a path with maximum score and write its score in the rectangle.

`64` `35` `40` `44`



Automation

Professor Zarbi wants to automate the computation of the maximum score of a number pyramid.
In his programs, a pyramid **Pn** is represented by a list `P` of `n` sub-lists numbered from `0` to `n-1`.
Each sub-list contains the numbers of one row.
The first sub-list `P[0]` contains the number at the top of the pyramid.
The last sub-list `P[n-1]` contains the `n` numbers of the last row.

Example

The pyramid **P5** shown here is displayed with its row numbers on the left.
For this pyramid:

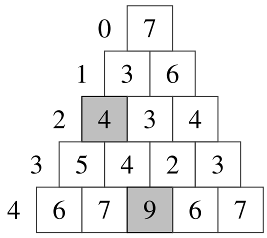
`n=5`

`P=[[7],[3,6],[4,3,4],[5,4,2,3],[6,7,9,6,7]]`

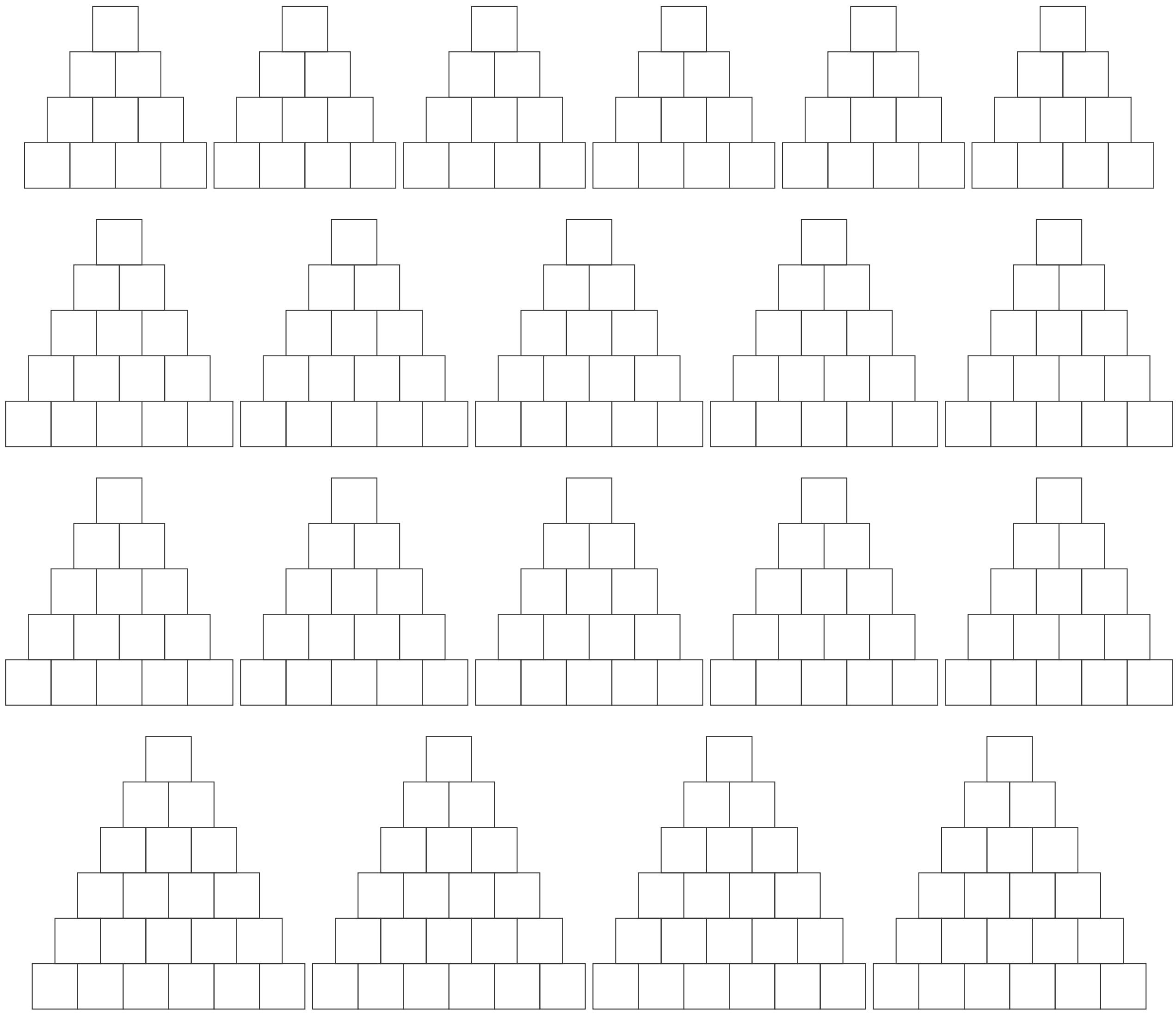
`P[0]=[7]`

`P[4]=[6,7,9,6,7]`

`P[2][0]=4` and `P[4][2]=9` are shaded in grey.



Pyramids for your draft work.



Function `MaxScore(n,P)`

This function uses 2 parameters.

- `n`: the number of rows of the pyramid.
- `P`: a list of lists of numbers representing the pyramid, as explained above.

The function returns the maximum possible score for a path in this pyramid.

Example: `MaxScore(5,[[7],[3,6],[4,3,4],[5,4,2,3],[6,7,9,6,7]])` returns `29` (see the first example on the previous page).

For each tile, the program computes a score equal to the maximum score of a path in the sub-pyramid having that tile as its top.
The score of a tile can be computed using the scores of the tiles it touches in the next row.
The score of a tile replaces in `P` the number written on the tile.

Example

During the processing of the left pyramid,
the function replaces the values in `P` with those shown in the right pyramid.

Left pyramid:

7				
3	6			
4	3	4		
5	4	2	3	
6	7	9	6	7

Right pyramid:

29				
20	22			
17	16	15		
12	13	11	10	
6	7	9	6	7

Before execution, `P=[[7],[3,6],[4,3,4],[5,4,2,3],[6,7,9,6,7]]`
After execution, `P=[[29],[20,22],[17,16,15],[12,13,11,10],[6,7,9,6,7]]`

The following question is worth between 0 and 10 points.
You lose 2 points for each zone ▭ that is not correctly completed. There are no negative points.

Q3(b) /10	Complete the ▭ in the function `MaxScore(n,P)`.

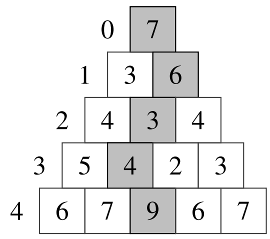
```
function MaxScore(n,P) {
  for r ← n-2 to 0 step -1
  {
    for c ← 0 to r step 1
    {
      if P[r+1][c] > P[r+1][c+1]

        {P[r][c] ← P[r][c] + P[r+1][c]}
      else

        {P[r][c] ← P[r][c] + P[r+1][c+1]}
    }
  }
  return P[0][0]
}
```

Function `GoodPath(n,P)`

What really matters is to find a path with maximum score in order not to trigger any traps.
A path is represented by a list that contains, for each row, the number of the tile on which the path passes in that row.

Example

The pyramid **P5** shown here is displayed with its row numbers on the left.
The path shaded in grey in the pyramid is described by the list `GP` of tile numbers within their row.
In this example, `GP=[0,1,1,1,2]`
Indeed, the path passes through tile `0` of row `0`,
tile `1` of rows `1`, `2` and `3`
and finally tile `2` of row `4`.



The function `GoodPath(n,P)` uses the same parameters as the function `MaxScore(n,P)`.
It returns a list that describes a path with maximum score in `P`.
It suffices to replace the **`return`** of the function `MaxScore(n,P)` with a few lines of code.

This question is worth between 0 and 6 points.
You lose 2 points for each zone ▭ that is not correctly completed. There are no negative points.

Reminder: `GP←[0]*n` creates a list of `n` zeros.

Q3(c) /6	Complete the ▭ in the code of **`GoodPath`** that replaces **`return`** in **`MaxScore`**.

```
c ← 0
GP ← [0]*n
for r ← 1 to n-1 n-1 1
{
  if P[r][c] < P[r][c+1]

    { c ← c+1 }

  GP[r] ← c
}
return GP
```