be-OI 2026 **Finale - JUNIOR** samedi 14 mars 2026	**Remplissez ce cadre en MAJUSCULES et LISIBLEMENT svp** PRÉNOM : ..... NOM : ..... ÉCOLE :	O **Réservé**

Finale de l'Olympiade belge d'Informatique 2026 (durée : **2h**)

Consignes à lire attentivement avant l'épreuve.

1. **Attendez le signal du départ** avant d'enlever les feuilles de la pochette en plastique.
2. Vous devez passer la finale dans la même langue que celle dans laquelle vous avez passé la qualification. **Vérifiez que cette page est dans la bonne langue.** Si ce n'est pas le cas, signalez-le ou changez de place.
3. Vérifiez que vous avez reçu la **bonne série de questions** mentionnée ci-dessus dans l'en-tête.
   - Pour les élèves jusqu'en deuxième année du secondaire: catégorie **cadet**, feuilles **roses**.
   - Pour les élèves en troisième ou quatrième année du secondaire: catégorie **junior**, feuilles **vertes**.
   - Pour les élèves de cinquième année du secondaire et plus: catégorie **senior**, feuilles **jaunes**.
4. Quand le signal de départ est donné, sortez les feuilles de la pochette, mais **n'enlevez pas les agrafes.**
5. Indiquez votre nom, prénom et école **très lisiblement en MAJUSCULES et uniquement sur cette page**.
6. Indiquez vos **réponses sur les feuilles de couleur** agrafées avec cette page.
   Écrivez de façon **bien lisible** à l'aide d'un **bic ou stylo** bleu ou noir.
7. Utilisez les feuilles blanches comme feuilles de brouillon.
   N'oubliez pas de recopier vos solutions sur les feuilles de couleur.
8. L'épreuve dure 2 heures et vous devez rester sur place jusqu'à la fin.
   Si vous avez terminé plus tôt, appelez un surveillant pour remettre vos réponses.
9. Quand le signal de fin est donné vous devez immédiatement remettre vos solutions.
   - Ne replacez pas les feuilles dans la pochette plastique.
   - **Déposez les feuilles de couleur, toujours agrafées**, dans une des boîtes prévues à cet effet.
   - **Rendez la pochette en plastique** en la déposant dans une boîte prévue à cet effet.
   - Gardez les feuilles blanches.
10. Vous ne pouvez avoir que de quoi écrire avec vous. Les calculatrices, smartphones, ... sont **interdits**.
11. Tous les extraits de code des énoncés sont en **pseudo-code**. Vous trouverez, sur les pages suivantes, une **description** du pseudo-code que nous utilisons. Si vous devez répondre en code, vous pouvez utiliser le **pseudo-code** ou un **langage de programmation courant** (Java, C, C++, Pascal, Python, ...). Les erreurs de syntaxe ne sont pas prises en compte pour l'évaluation.

L'Olympiade Belge d'Informatique est possible grâce au soutien de nos membres:

Aide-mémoire pseudo-code

Les données sont stockées dans des variables, chacune est identifiée par un nom.
Le nom permet d'accéder à la variable pour placer ou récupérer les données.
On utilise ← comme opérateur d'affectation pour stocker une valeur dans une variable.
Exemple : `n←100` place le nombre entier `100` dans la variable dénommée `n`.

Variables et données simples	Exemples
nombre entier	`n←100`
	`m←-1`
nombre réel	`pi←3.1415`
	`z←0.0`
valeur logique (booléen)	`t←`**`true`** (vrai)
	`t←`**`false`** (faux)

On peut effectuer des opérations arithmétiques en utilisant des nombres et des variables.

addition et soustraction avec + et − (exemple : `a+3`)
multiplication avec ⋆ ou × (exemple : `2⋆a⋆b`)
division entière : quotient avec / ou // et reste avec % (exemple : `14//3` est égal à `4` et `14%3` est égal à `2`)
division non entière / : peu utilisée, on signale si c'est le cas (exemple : `3.6/4.0` est égal à `0.9`)
puissance ou exposant avec ^ (exemple : `x^3` est égal à `x⋆x⋆x`)

On utilise souvent des variables pour calculer un résultat
et le placer ensuite dans une variable.
Parfois le résultat est placé dans une des variables utilisées dans le calcul.
Le cadre à droite contient du code qui illustre cela.
Après l'exécution de ce code, `x=25`, `a=4` et `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
```

L'instruction **`if`** permet d'exécuter du code uniquement si une condition est vraie.
On peut éventuellement ajouter l'instruction **`else`** pour exécuter un autre code uniquement si la condition est fausse.
Dans l'exemple à gauche, si le nombre dans la variable `a` est plus petit que celui dans la variable `b`, alors on ajoute `5` dans la variable `p`,
sinon on soustrait `2` dans la variable `p`.
Ensuite, dans tous les cas, on enlève `1` dans la variable `c`.

Les instructions à exécuter dans les différents cas seront clairement identifiées
soit en les plaçant entre accolades { }, soit en les décalant.
Bien souvent, on fera les deux : mise entre accolades et décalage, comme ci-dessus.

Voici la liste des opérateurs de comparaison les plus courants.

= ou ==	<	<= ou ≤	>	>= ou ≥	! = ou ≠
est égal à	plus petit que	plus petit ou égal	plus grand que	plus grand ou égal	est différent de

On peut tester plusieurs conditions en les combinant avec les opérateurs logiques **`and`**, **`or`**, **`not`**.

La condition (p and q) est vraie si les 2 conditions `p` et `q` sont vraies.
La condition (p or q) est vraie si au moins une des 2 conditions est vraie.
La condition not(p) est vraie si `p` est fausse et fausse si `p` est vraie.
On peut placer la valeur logique d'une condition dans une variable booléenne à utiliser plus tard. Exemple : `f←((a==5) and (b>=0)) or ((a<0) and (b!=10))`
La condition d'un **`if`** est parfois une simple variable booléenne. Exemple : **`if`** `(f) {a←10}` **`else`** `{b←10}`

Il faut parfois stocker plusieurs données dans une seule variable structurée.

Variables et données structurées	Exemples
tableau, liste, vecteur	`seq←[7,11,0,-4,9]`
tableau, matrice	`M←[[0,1,2],[3,-1,3],[0,0,5]]`
couple	`coord←(1,7)`
texte	`n←"John"`

Les éléments individuels d'une variable structurée sont repérés par un indice que l'on écrit entre crochets après le nom de la variable.
Le premier élément d'une variable structurée `G` est d'indice `0` et est noté `G[0]`.
Le second élément est d'indice `1` et le dernier est d'indice `n-1` s'il y a en tout `n` éléments.
Le nombre d'éléments d'une variable structurée est retourné par la fonction `len()`.
Exemple : si `G=[5,9,12]` alors `len(G)=3`, `G[0]=5`, `G[1]=9` et `G[2]=12`.
Le tableau est de taille `3`, mais l'indice le plus élevé est `2`.

Pour répéter du code, par exemple pour parcourir les éléments d'un tableau, on peut utiliser une boucle **`for`**.

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

La notation **`for`** `(i←a to b step k)`
représente une boucle

- dans laquelle `i` commence à la valeur `a`,
- qui sera répétée tant que $i \leq b$,
- qui augmente `i` de `k` à la fin de chaque étape.

L'exemple à gauche calcule la somme des éléments du tableau `T`.
Les éléments sont ajoutés un par un dans la variable `sum`.

Les instructions à exécuter dans une boucle seront clairement identifiées soit en les plaçant entre accolades { },
soit en les décalant. Bien souvent, on fera les deux : mise entre accolades et décalage, comme ci-dessus.

On peut également écrire une boucle à l'aide de l'instruction **`while`** qui répète du code tant que sa condition est vraie.

Dans l'exemple à droite, on divise un nombre entier positif `n` par `2`,
puis par `3`, ensuite par `4` ...
jusqu'à ce qu'il ne soit plus composé que d'un seul chiffre
(c'est-à-dire jusqu'à ce que `n<10`).

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

Question 1 – Le palais perdu épisode 1 : compter.

Des archéologues ont découvert un ancien palais avec de nombreuses salles.
Mais il y a des pièges qu'ils ne parviennent pas à éviter.
Ils ont appelé le professeur Zarbi à la rescousse.

Le professeur a vite compris qu'il fallait marcher sur certaines dalles gravées pour désactiver les pièges.
Dans les salles, les dalles gravées sont toujours disposées en plusieurs rangées horizontales.
La première rangée est constituée d'une seule dalle gravée, appelée le *sommet*.

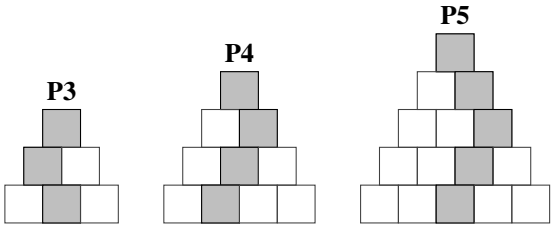
Le professeur a découvert qu'il faut avancer en respectant certaines règles, sinon un piège se déclenche inévitablement.

1. Il faut démarrer du sommet (en haut sur les exemples) et marcher uniquement sur des dalles gravées.
2. En avançant, on doit toujours passer d'une dalle gravée à une autre qui la touche (pas seulement par un coin).
3. Il faut passer par une seule dalle gravée dans chaque rangée.
4. Il faut atteindre la dernière rangée de dalles gravées (tout en bas sur les exemples).

À partir de maintenant, le mot ***chemin*** signifie une traversée qui respecte ces 4 règles.
À partir de maintenant, toutes les ***dalles*** citées ou dessinées sont des dalles gravées.

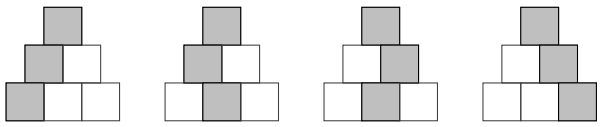
Les dalles sont souvent disposées en pyramide, comme dans tous les exemples sur cette page.
On note **Pn** une pyramide constituée de n rangées de dalles.

Exemple 1
Voici des pyramides **P3**, **P4** et **P5**.
Un exemple de chemin est colorié en gris dans chacune.
Le sommet est en gris (règle 1).
Les dalles grises de rangées adjacentes se touchent (règle 2).
Il y a une seule dalle grise par rangée (règle 3).
Il y a une dalle grise dans la dernière rangée (règle 4).



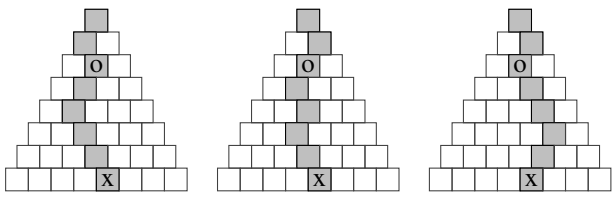
Le professeur Zarbi veut compter combien de chemins différents permettent de traverser des pyramides.

Exemple 2
Voici tous les chemins qui traversent **P3**.
1 chemin atteint la dalle de gauche de la dernière rangée.
2 chemins atteignent la dalle au centre de la dernière rangée.
1 chemin atteint la dalle de droite de la dernière rangée.
Au total, 4 chemins différents traversent **P3**.



Parfois, il veut compter uniquement les chemins qui passent par certaines dalles.

Exemple 3
Voici 3 chemins qui traversent **P8**.
Ils passent tous par la deuxième dalle de la troisième rangée (dalle marquée d'un o).
Ils atteignent tous la cinquième dalle de la dernière rangée (dalle marquée d'un x).

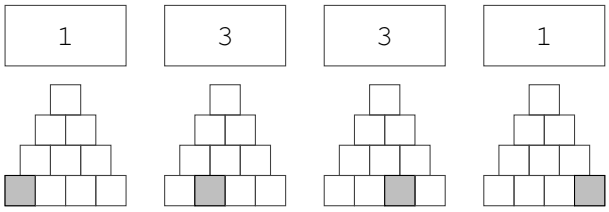


Remarque : sur une des dernières pages de ce questionnaire, des pyramides vides sont à votre disposition pour vos recherches au brouillon.

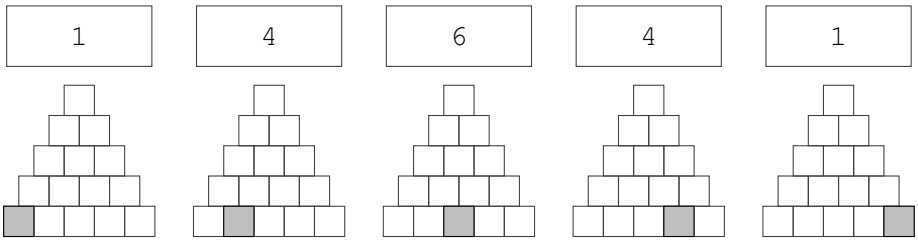
Compter les chemins qui atteignent une dalle précise.

Dans les questions suivantes, des pyramides sont dessinées avec 1 dalle grise dans la dernière rangée.
Combien de chemins différents atteignent la dalle grise ? Écrivez les réponses dans les rectangles.

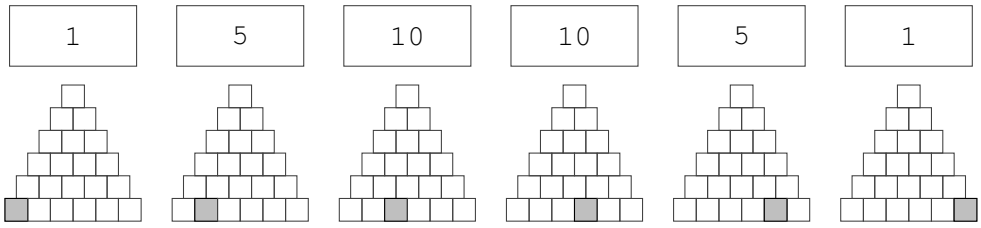
Q1(a) /4	**Combien de chemins atteignent la dalle grise dans P4 ?**



Q1(b) /5	**Combien de chemins atteignent la dalle grise dans P5 ?**

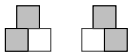


Q1(c) /6	**Combien de chemins atteignent la dalle grise dans P6 ?**

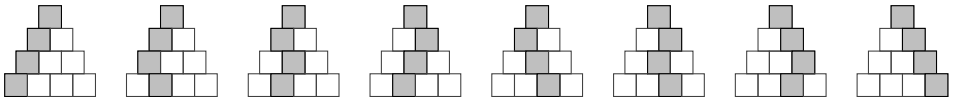


Compter tous les chemins.

Q1(d) /1	**Au total, combien de chemins traversent P2 ? Solution : 2**



Q1(e) /1	**Au total, combien de chemins traversent P4 ? Solution : 8**



Q1(f) /1	**Au total, combien de chemins traversent P5 ? Solution : 16**

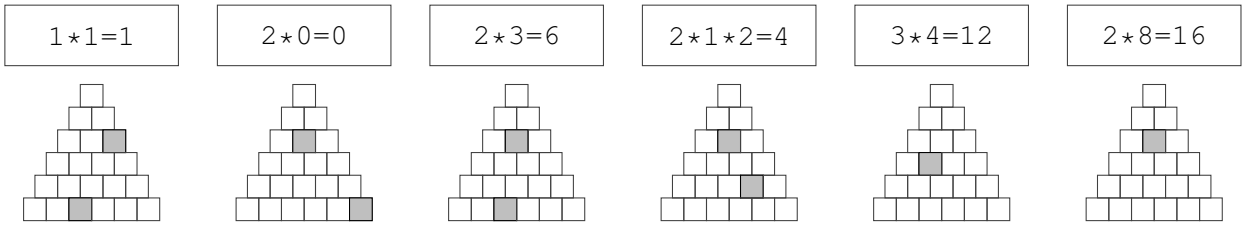
Q1(g) /1	**Au total, combien de chemins traversent P6 ? Solution : 32**

Q1(h) /1	**Au total, combien de chemins traversent Pn (où n est un nombre entier positif) ? Solution :** 2^{n-1}

Compter les chemins qui passent par certaines dalles.

Dans les questions suivantes, des pyramides sont dessinées avec une ou plusieurs dalles grises.
Écrivez dans le rectangle au-dessus de la pyramide le nombre de chemins différents qui passent par les dalles grises.
N'oubliez pas les 4 règles !

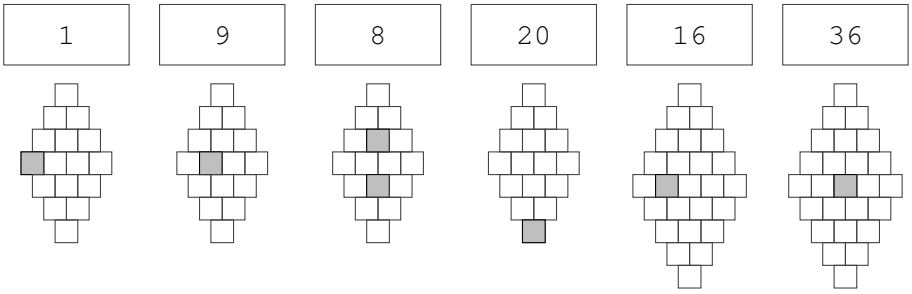
Q1(i) /6	Combien de chemins passent par les dalles grises des pyramides P6 ?



Dalles disposées en diamant.

Les dalles sont souvent disposées en pyramide, mais ce n'est pas toujours le cas.
Il existe par exemple une disposition en diamant où la dernière rangée n'a qu'une seule dalle.
Dans les questions suivantes, des salles en diamant sont dessinées avec une ou plusieurs dalles grises.
Écrivez dans le rectangle au-dessus du diamant le nombre de chemins différents qui passent par les dalles grises.

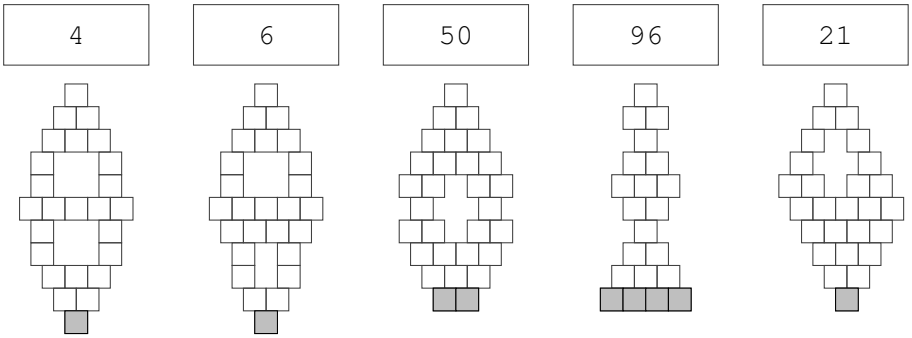
Q1(j) /6	Combien de chemins passent par les dalles grises des diamants ?



Salles étranges.

Les dalles de certaines salles sont disposées de manière compliquée.
Dans les questions suivantes, il faut compter tous les chemins qui traversent des salles étranges.
Écrivez dans le rectangle au-dessus de la salle le nombre total de chemins qui la traversent (jusqu'à une dalle grise).

Q1(k) /5	Au total, combien de chemins traversent ces salles étranges ?



Étapes dans les grandes pyramides.

Certaines salles contiennent de très grandes pyramides de dalles et le professeur Zarbi veut trouver un moyen de simplifier les comptages de chemins.
Il utilise les conventions ci-dessous.

- Les `n` rangées de dalles d'une pyramide **Pn** sont numérotées de `0` à `n-1` en partant d'en haut.
- Dans une pyramide, la rangée numéro `r` a toujours `r+1` dalles numérotées de `0` à `r` en partant de la gauche.
- Les coordonnées d'une dalle sont `(r,c)` où `r` est sa rangée et `c` son numéro dans sa rangée.

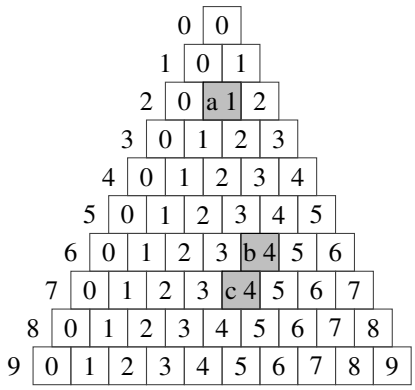
Exemple

La figure ci-contre représente une pyramide **P10**.
Les numéros de rangées sont affichés à gauche.
Les numéros utilisés dans chaque rangée sont affichés sur les dalles.
Les 3 dalles grises sont repérées avec les lettres a, b et c.
Leurs coordonnées sont `(2,1)`, `(6,4)` et `(7,4)`.

Remarque

Cette figure donne aussi les coordonnées des dalles dans les plus petites pyramides.
Par exemple, les 4 premières rangées montrent les coordonnées dans une pyramide **P4**.

P10



Le professeur Zarbi compte en plusieurs étapes les chemins passant par des dalles grises d'une grande pyramide.
Chaque étape utilise une sous-pyramide.

- La première sous-pyramide a le même sommet que la grande pyramide, et sa dernière rangée contient la première dalle grise de la grande pyramide.
- Ensuite, on utilise des sous-pyramides dont le sommet est une dalle grise, et dont la dernière rangée contient la dalle grise suivante.
- Il y a un cas particulier si la dernière rangée de la grande pyramide ne contient pas de dalle grise. Dans ce cas, il n'y a pas de dalle grise dans la dernière rangée de la dernière sous-pyramide.

Le professeur Zarbi utilise les notations suivantes.

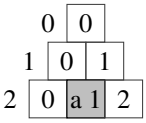
- `C(r,c)` est le nombre de chemins atteignant la dalle `(r,c)` (dans une pyramide à `r+1` rangées).
- `T(n)` est le nombre total de chemins qui traversent une pyramide **Pn** (donc à `n` rangées).
  Attention : pour les besoins de cette finale **BeOI**, il est **interdit d'utiliser `T(1)`**.

Exemple

Voici la décomposition en 4 étapes du comptage des chemins dans **P10** ci-dessus.
Première étape : du sommet à la première dalle grise a.
Deuxième étape : de la première dalle grise a à la deuxième b.
Troisième étape : de la deuxième dalle grise b à la troisième c.
Dernière étape : de la dernière dalle grise c à la dernière rangée de **P10**.
Le nombre de chemins possibles pour chaque étape est noté dans le rectangle au-dessus de l'étape.

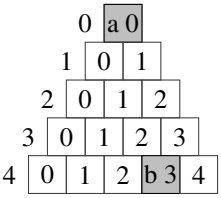
`C(2,1)`

P3



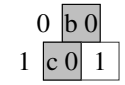
`C(4,3)`

P5



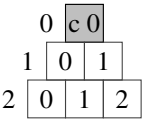
`C(1,0)`

P2



`T(3)`

P3

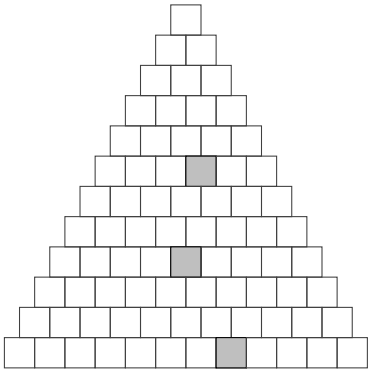


Évidemment, il faut utiliser les nombres de chemins possibles des étapes pour déterminer le nombre de chemins passant par les dalles grises dans la grande pyramide.
Dans les questions suivantes, donnez une expression mathématique utilisant les notations `C(r,c)` et `T(n)` pour déterminer le nombre de chemins qui passent par les dalles grises de l'exemple **P10** de la page précédente et des pyramides **P12**, **P13** et **P15** ci-dessous.

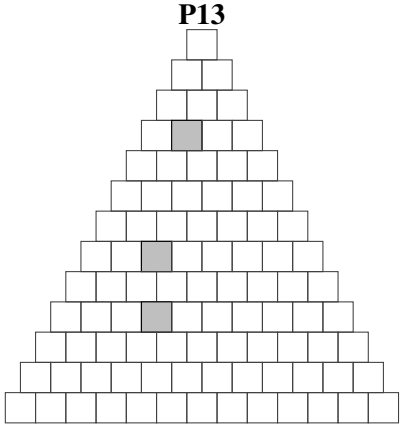
Remarque importante
Ne calculez pas la réponse finale !
Vous devez écrire une expression mathématique ressemblant par exemple à `C(5,2)*T(3)+C(4,2)`.

Q1(l) /2	**Expression du nombre de chemins passant par les dalles grises de P10 de l'exemple.** **Solution :** `C(2,1) * C(4,3) * C(1,0) * T(3)`
Q1(m) /2	**Expression du nombre de chemins passant par les dalles grises de P12.** **Solution :** `C(5,3) * C(3,1) * C(3,3)`
Q1(n) /2	**Expression du nombre de chemins passant par les dalles grises de P13.** **Solution :** `C(3,1) * C(4,1) * C(2,1) * T(4)`
Q1(o) /2	**Expression du nombre de chemins passant par les dalles grises de P15.** **Solution :** `C(3,2) * C(4,2) * C(3,0) * C(3,1) * T(2)`

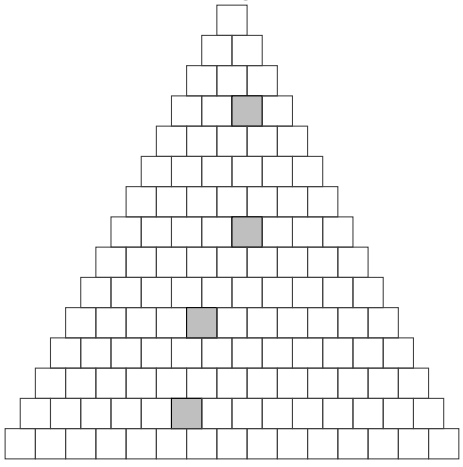
P12



P13



P15



Automatisation.

Le professeur Zarbi a écrit une fonction `countPyra(n,G)` pour automatiser le comptage du nombre de chemins passant par des dalles grises dans une grande pyramide.

Cette fonction utilise 2 paramètres correctement initialisés.

- `n` : un entier qui contient le nombre de rangées de la grande pyramide.
- `G` : un tableau de couples `(r,c)` qui contient les coordonnées des dalles grises (dans l'ordre de haut en bas).

Le programme utilise aussi les fonctions `C(r,c)` et `T(n)` qui retournent les valeurs expliquées précédemment.
Enfin, le programme utilise une boucle "**`for`** `(r,c)` **`in`** `G`" qui s'exécute autant de fois qu'il y a d'éléments dans `G`.

Exemples avec la pyramide P10 dessinée plus haut dans ce document.

- `n=10`
- `G=[(2,1), (6,4), (7,4)]`
  Donc `G[0]=(2,1)`, `G[1]=(6,4)`, `G[2]=(7,4)`.
- `(r0,c0)←(0,0)` initialise les 2 variables en même temps (c'est équivalent à `r0←0` et `c0←0`).
- `C(2,1)` retourne `2` et `T(3)` retourne `4`.
- La boucle "**`for`** `(r,c)` **`in`** `G`" s'exécute 3 fois.
  La première fois `(r,c)=(2,1)`, la seconde fois `(r,c)=(6,4)`, la dernière fois `(r,c)=(7,4)`.

Le professeur Zarbi écrit une première version du programme qui ne fait aucune vérification.
Ce programme simple fonctionne correctement avec les 4 grandes pyramides **P10**, **P12**, **P13** et **P15** à condition d'initialiser convenablement `n` et `G`.

Le programme utilise les variables suivantes.

- `nbr` : le nombre de chemins passant par les dalles grises de la grande pyramide.
- `h` : le numéro de la dernière rangée de la sous-pyramide en cours de traitement (qui a donc `h+1` rangées).
- `k` : le numéro de la dalle grise dans la dernière rangée de la sous-pyramide en cours (avec $0 \leq k \leq h$).

La question suivante rapporte entre 0 et 6 points.
Vous perdez 2 points pour chaque zone ____ qui n'est pas correctement complétée. Il n'y a pas de points négatifs.

Q1(p) /6	**Complétez les ____ dans la fonction `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
}
```

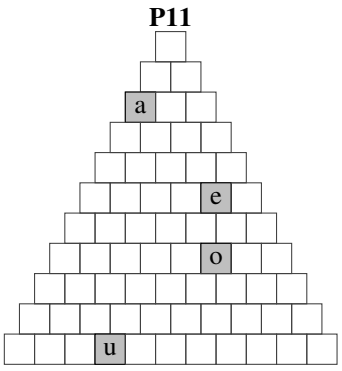
Certaines pyramides peuvent poser problème.

Exemple

Il n'y a pas de chemins passant par toutes les dalles grises de la pyramide ci-contre.

Il est impossible d'aller de a à e.

De même, il n'y a pas de sous-pyramide de sommet o dont la dernière rangée contient u.



Le programme `countPyraV1` ne tient pas compte de ces cas et peut donner de mauvaises réponses ou s'arrêter avec un message d'erreur !
Le programme `countPyraV2` est une amélioration qui donne la bonne réponse dans tous les cas.
Le professeur Zarbi a seulement remplacé la ligne de code `nbr←` ____ de `countPyraV1` par un test dans `countPyraV2`.
Quelles sont les conditions à vérifier et que faut-il faire suivant qu'elles sont vraies ou qu'elles sont fausses ?
À vous d'écrire le test dans la question suivante.

Cette question rapporte ente 0 et 6 points.
Vous perdez 2 points pour chaque zone ____ qui n'est pas correctement complétée. Il n'y a pas de points négatifs.

Q1(q) /6	**Complétez les ____ dans le test de `countPyraV2` qui remplace une ligne de `countPyraV1`.**

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

Question 2 – Le palais perdu épisode 2 : décrypter.

Les découvertes du professeur Zarbi ne permettent pas encore d'éviter tous les pièges.
Mais les archéologues ont découvert les plans des salles parmi d'autres documents importants.
Dans ces plans, une pyramide marquée de petits disques est dessinée sur chaque dalle.
Le professeur comprend que ces pyramides représentent des nombres et il parvient à les décrypter.

Attention ! Ne confondez pas !

Dans l'épisode 1, les pyramides étaient constituées de dalles.

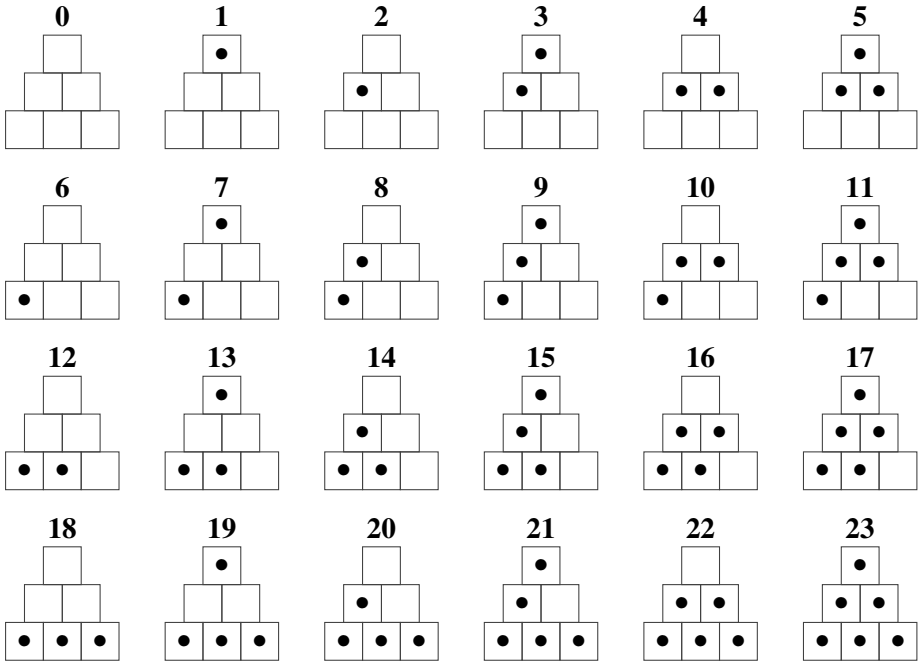
Dans l'épisode 2, les pyramides sont des nombres marqués sur les dalles.

Le système de numérotation.

Les nombres sont représentés par des pyramides dont certaines cases contiennent un petit disque noir.

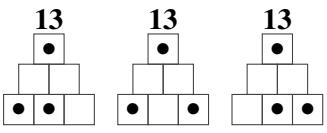
- 0 est représenté par une pyramide sans aucun disque.
- Pour passer d'une pyramide à la suivante, c'est-à-dire d'un nombre au suivant, on procède comme suit.
  1. En partant du sommet, on cherche la première rangée qui contient une case vide.
  2. On place un petit disque dans n'importe quelle case vide de cette rangée.
  3. On vide toutes les cases des rangées supérieures (celles qui étaient remplies).

Exemples : voici des pyramides **P3** qui représentent les nombres de 0 à 23.



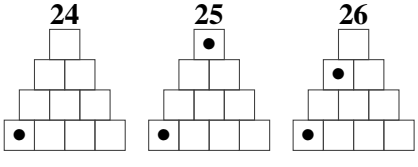
Nombres qui ont plusieurs représentations.

Plusieurs pyramides peuvent représenter le même nombre puisqu'on peut placer les disques noirs n'importe où dans leur rangée (point 2 ci-dessus).
Exemple : les 3 pyramides ci-contre représentent toutes le nombre 13.



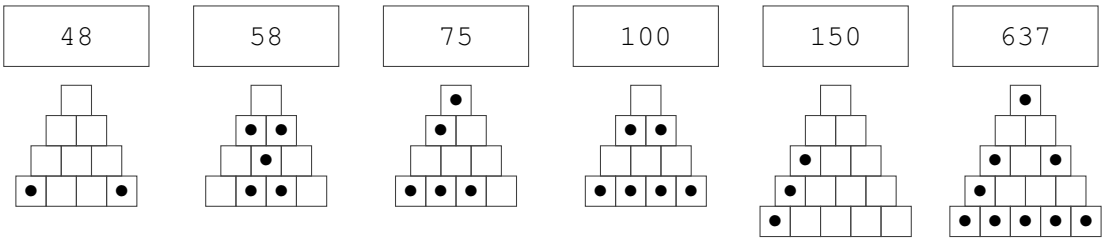
Représentations de nombres plus grands.

23 est le plus grand nombre qu'on peut représenter par une pyramide **P3**.
Pour aller plus loin, il faut utiliser des pyramides avec plus de rangées.

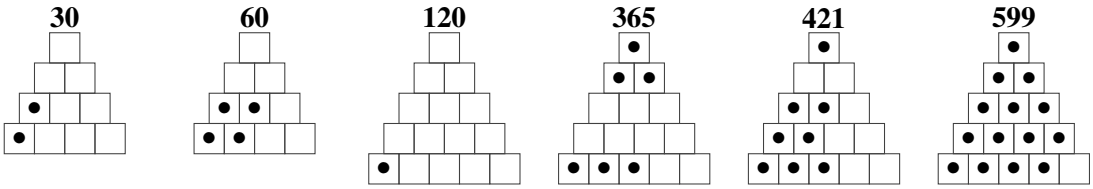


Remarque : ***dessiner une pyramide*** signifie ***dessiner des petits disques noirs*** **dans certaines cases de celle-ci**.
Cas particulier : si on ne dessine aucun disque, on *dessine une pyramide vide* qui représente le nombre 0.

Q2(a) /6	**Écrivez dans chaque rectangle le nombre représenté par la pyramide dessinée.**



Q2(b) /6	**Dessinez une pyramide qui représente le nombre indiqué.**



Plus haut, on a fait remarquer que plusieurs pyramides peuvent représenter le même nombre et que 23 est le plus grand qu'on peut représenter par une pyramide **P3**. Cela inspire les questions suivantes auxquelles vous pouvez répondre par un nombre ou par une expression mathématique.

Q2(c) /1	**Combien de pyramides P3 différentes peut-on dessiner ?** Solution : $2^6 = 64$

Q2(d) /1	**Combien de pyramides P4 différentes peut-on dessiner ?** Solution : $2^{10} = 1024$

Q2(e) /1	**Quel est le plus grand nombre qu'on peut représenter avec une pyramide P4 ?** Solution : $5! - 1 = 119$

Q2(f) /1	**Combien de pyramides P5 différentes peut-on dessiner ?** Solution : $2^{15} = 32768$

Q2(g) /1	**Quel est le plus grand nombre qu'on peut représenter avec une pyramide P5 ?** Solution : $6! - 1 = 719$

Q2(h) /1	**Combien de pyramides P6 différentes peut-on dessiner ?** Solution : $2^{21} = 2097152$

Q2(i) /1	**Quel est le plus grand nombre qu'on peut représenter avec une pyramide P6 ?** Solution : $7! - 1 = 5039$

Dans les questions suivantes, vous devez donner tous les nombres qui ont une certaine propriété.
S'il y a plusieurs nombres, écrivez-les en les séparant par des virgules.
S'il n'y a pas de nombre, répondez en traçant une croix ×.

Q2(j) /1	**Quels nombres ont une seule représentation dans P3 ?** Solution : 0, 1, 4, 5, 18, 19, 22, 23

Q2(k) /1	**Quels nombres ont exactement 2 représentations dans P3 ?** Solution : 2, 3, 20, 21

Q2(l) /1	**Quels nombres ont exactement 3 représentations dans P3 ?** Solution : 6, 7, 10, 11, 12, 13, 16, 17

Q2(m) /1	**Quels nombres ont exactement 4 représentations dans P3 ?** Solution : X

Q2(n) /1	**Quels nombres ont exactement 5 représentations dans P3 ?** Solution : X

Q2(o) /1	**Quels nombres ont exactement 6 représentations dans P3 ?** Solution : 8, 9, 14, 15

Q2(p) /1	**Quels nombres ont exactement 36 représentations dans P4 ?** Solution : 56, 57, 62, 63

Q2(q) /1	**Quel est le plus petit nombre qui a exactement 10 représentations dans P5 ?** Solution : 122

Q2(r) /1	**Quel est le plus grand nombre qui a exactement 10 représentations dans P5 ?** Solution : 597

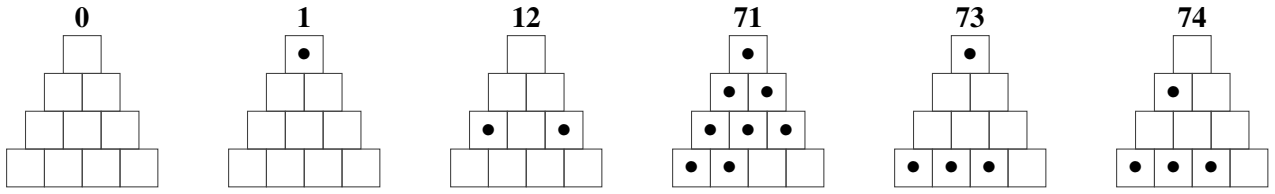
Nouvelle notation : liste d'une pyramide.

Dessiner les pyramides est fastidieux et peu pratique.
Au lieu de dessiner une pyramide, le professeur Zarbi préfère donner la liste, en partant du sommet, des nombres de disques noirs dans chaque rangée.

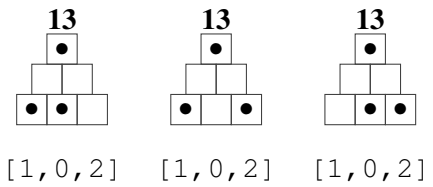
À partir de maintenant, le ***nombre d'une pyramide*** est le nombre représenté par la pyramide.
C'est un nombre entier positif, noté comme d'habitude avec les chiffres de 0 à 9.

À partir de maintenant, la ***liste d'une pyramide*** est la liste des nombres de disques noirs dans chaque rangée de la pyramide.
Cette liste commence par `0` s'il n'y a pas de disque noir au sommet ou par `1` s'il y en a un.
Cette liste contient `n` nombres pour une pyramide **Pn**.

Exemples : voici quelques pyramides **P4**.
Pour chacune, son **nombre** est indiqué au-dessus et sa **liste** en-dessous.



Remarque.
Si plusieurs pyramides représentent le même nombre, alors elles ont la même liste.
Exemple : les 3 pyramides **P3** qui représentent le nombre 13 ont la même liste `[1,0,2]`.



Automatisation.

Le professeur doit souvent calculer le nombre d'une pyramide à partir de sa liste.
Il lui arrive aussi, inversement, de calculer la liste d'une pyramide à partir de son nombre.
Il veut écrire des programmes pour automatiser ses calculs.

Quelques rappels.

- **Division entière.**
  L'opérateur `/` (ou `//` si vous préférez) donne le quotient entier d'une division entre 2 entiers.
  L'opérateur `%` donne le reste d'une division entre 2 entiers.
  Exemple : `14/3` est égal à `4` (de même que `14//3`) et `14%3` est égal à `2`.
- **Numérotation des éléments d'une liste.**
  Les éléments d'une liste sont numérotés à partir de `0`.
  Exemple : si `LIS=[1,0,2]` alors `LIS[0]=1`, `LIS[1]=0` et `LIS[2]=2`.
- **Initialisation d'une liste.**
  `LIS←[0]*n` crée une liste de `n` zéros.

Fonction `PyrLIS(n,dec)`
Cette fonction reçoit 2 paramètres.

- `n` : le nombre de rangées de la pyramide.
- `dec` : un nombre entier positif qu'il faut représenter par une pyramide.
  `dec` est nommé ainsi pour rappeler qu'il s'agit d'un nombre **décimal** (c'est-à-dire écrit avec les 10 chiffres de 0 à 9).

La fonction retourne la liste d'une pyramide **Pn** qui représente le nombre `dec`.

Exemples : `PyrLIS(3,13)` retourne `[1,0,2]` et `PyrLIS(4,13)` retourne `[1,0,2,0]`.

La question suivante rapporte entre 0 et 6 points.
Vous perdez 3 points pour chaque zone ____ qui n'est pas correctement complétée.

Q2(s) /6	Complétez les ____ dans la fonction `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)
}
```

Fonction `Pyrdec(n,LIS)`
Cette fonction reçoit 2 paramètres.

- `n` : le nombre de rangées de la pyramide.
- `LIS` : la liste d'une pyramide.

La fonction retourne le nombre `dec` de la pyramide **Pn** dont la liste est donnée.

Exemples : `Pyrdec(3,[1,0,2])` et `Pyrdec(4,[1,0,2,0])` retournent `13`.

La question suivante rapporte entre 0 et 6 points.
Vous perdez 3 points pour chaque zone ____ qui n'est pas correctement complétée.

Q2(t) /6	Complétez les ____ dans la fonction `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 – Le palais perdu épisode 3 : additionner.

Le professeur Zarbi a décrypté tous les plans où de petites pyramides à disques étaient dessinées sur les dalles.
Il a recopié les plans en remplaçant chaque petite pyramide par le nombre qu'elle représente.
Après de longues réflexions et des expérimentations minutieuses, il a enfin trouvé la solution...

Attention ! Ne confondez pas !

Dans l'épisode 3, les pyramides sont à nouveau constituées de dalles, comme dans l'épisode 1.
Mais dans cet épisode, chaque dalle porte un nombre (celui décrypté dans l'épisode 2).

Chemins de somme maximale.

À partir de maintenant, une ***pyramide de nombres*** est une pyramide avec un nombre sur chaque dalle.

Pour rappel, un ***chemin***

- démarre du sommet,
- passe d'une dalle à une autre qui la touche,
- passe sur une seule dalle dans chaque rangée,
- doit atteindre la dernière rangée.

Le ***score*** d'un chemin est la somme des nombres marqués sur les dalles de ce chemin.
Le professeur Zarbi a découvert que le chemin qui a le plus grand score ne déclenche pas de piège.
Si plusieurs chemins ont le même score maximal, aucun ne déclenche de piège.
Tout chemin qui a un score inférieur au score maximal déclenche un piège.

Exemples : voici quelques pyramides **P5**.
Pour chacune, un chemin de score maximal est colorié en gris.
Le score du chemin est indiqué dans le rectangle.

`7+6+3+4+9=29`

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

`10+16+8+19+18=71`

		10		
	14	16		
9	8	7		
17	4	19	6	
15	20	15	18	3

`7+5+11+17+15=55`

		7		
	8	5		
10	9	11		
15	16	14	17	
11	12	13	15	14

`1+1+5+1+17=25`

		1		
	1	2		
5	3	1		
1	4	7	10	
17	13	9	5	1

Dans la question suivante, des pyramides sont dessinées avec un nombre sur chaque dalle.
Dans chacune, tracez un chemin de score maximal (en entourant les nombres ou en coloriant les dalles au crayon).
Écrivez le score du chemin tracé dans le rectangle au-dessus de la pyramide.

Q3(a) /8	**Tracez un chemin de score maximal et indiquez son score dans le rectangle.**

`64`

		11		
	11	18		
18	11	18		
24	23	15	14	

`35`

		7		
	9	9		
6	4	5		
4	6	7	5	
8	5	6	5	9

`40`

		7		
	3	6		
8	8	4		
9	7	3	4	
1	9	4	1	2
4	1	3	5	1	8

`44`

			7			
		6	2			
	7	4	5			
5	5	6	1			
6	6	7	7	4		
1	4	6	1	7	5	
2	5	1	6	5	6	3

Automatisation

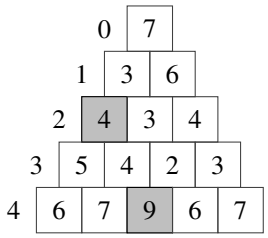
Le professeur Zarbi veut automatiser le calcul du score maximal d'une pyramide de nombres.
Dans ses programmes, une pyramide **Pn** est représentée par une liste `P` de `n` sous-listes numérotées de `0` à `n-1`.
Chaque sous-liste contient les nombres d'une rangée.
La première sous-liste `P[0]` contient le nombre au sommet de la pyramide.
La dernière sous-liste `P[n-1]` contient les `n` nombres de la dernière rangée.

Exemple

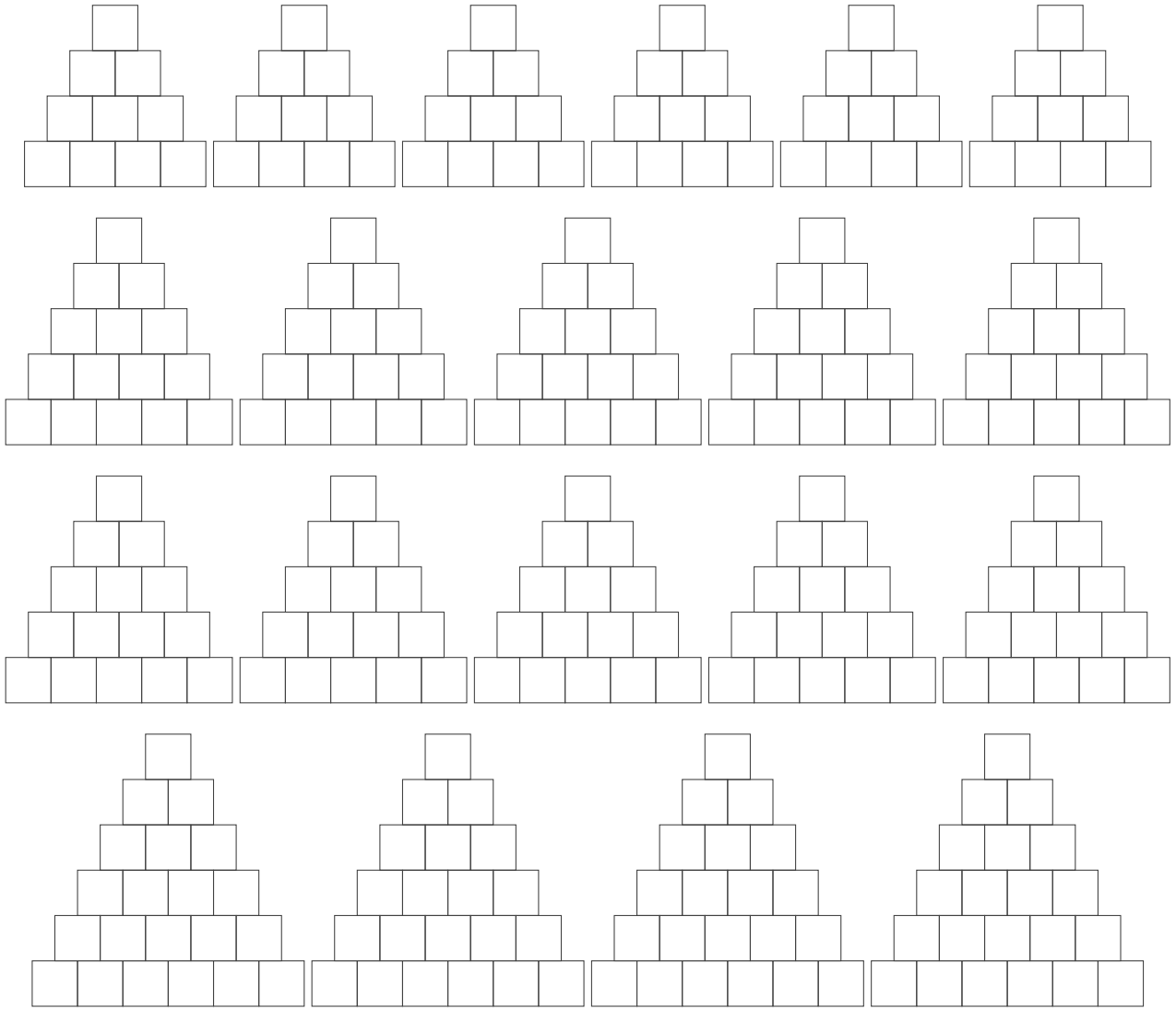
La pyramide **P5** ci-contre est affichée avec ses numéros de rangées à gauche.
Pour cette pyramide :

```
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` et `P[4][2]=9` sont coloriés en gris.



Pyramides à utiliser pour vos recherches au brouillon.



Fonction `MaxScore(n,P)`
Cette fonction utilise 2 paramètres.

- `n` : le nombre de rangées de la pyramide.
- `P` : une liste de listes de nombres qui représente la pyramide, comme expliqué ci-dessus.

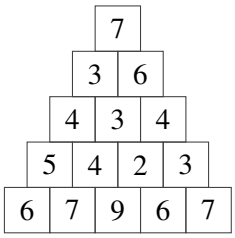
La fonction retourne le score maximal possible pour un chemin dans cette pyramide.

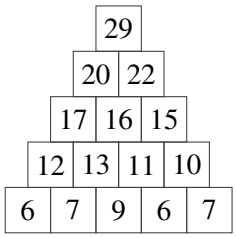
Exemple : `MaxScore(5,[[7],[3,6],[4,3,4],[5,4,2,3],[6,7,9,6,7]])` retourne `29` (voir le premier exemple de la page précédente).

Pour chaque dalle, le programme calcule un score égal au score maximal d'un chemin dans la sous-pyramide ayant cette dalle pour sommet.
On peut calculer le score d'une dalle en utilisant les scores des dalles qu'elle touche dans la rangée suivante.
Le score d'une dalle remplace dans `P` le nombre écrit sur la dalle.

Exemple

Pendant le traitement de la pyramide de gauche,
la fonction remplace les valeurs dans `P` par celles affichées dans la pyramide de droite.





Avant l'exécution, `P=[[7],[3,6],[4,3,4],[5,4,2,3],[6,7,9,6,7]]`
Après l'exécution, `P=[[29],[20,22],[17,16,15],[12,13,11,10],[6,7,9,6,7]]`

La question suivante rapporte entre 0 et 10 points.
Vous perdez 2 points pour chaque zone _____ qui n'est pas correctement complétée. Il n'y a pas de points négatifs.

Q3(b) /10	**Complétez les _____ dans la fonction `MaxScore(n,P)`.**

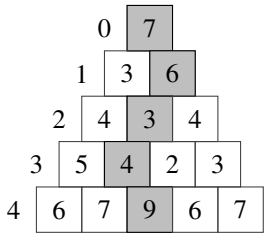
```
function MaxScore(n,P) {
  for r ← n-2 to 0 step -1
  {
    for c ← 0 to r step 1
    {
      if P[r+1][c] > P[r+1][c+1]

        {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]
}
```

Fonction `GoodPath(n,P)`

Ce qui compte vraiment, c'est de trouver un chemin de score maximal afin de ne pas déclencher de pièges.
Un chemin est représenté par une liste qui contient, pour chaque rangée, le numéro de la dalle sur laquelle le chemin passe dans cette rangée.

Exemple

La pyramide **P5** ci-contre est affichée avec ses numéros de rangées à gauche.
Le chemin colorié en gris dans la pyramide est décrit par la liste `GP` des numéros de dalles dans leur rangée.
Dans cet exemple, `GP=[0,1,1,1,2]`
En effet, le chemin passe par la dalle `0` de la rangée `0`,
la dalle `1` des rangées `1`, `2` et `3`
et enfin par la dalle `2` de la rangée `4`.



La fonction `GoodPath(n,P)` utilise les mêmes paramètres que la fonction `MaxScore(n,P)`.
Elle retourne une liste qui décrit un chemin de score maximal dans `P`.
Il suffit de remplacer le **`return`** de la fonction `MaxScore(n,P)` par quelques lignes de codes.

Cette question rapporte ente 0 et 6 points.
Vous perdez 2 points pour chaque zone ____ qui n'est pas correctement complétée. Il n'y a pas de points négatifs.

Rappel : `GP←[0]*n` crée une liste de `n` zéros.

Q3(c) /6	**Complétez les ____ dans le code de `GoodPath` qui remplace `return` dans `MaxScore`.**

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

    { c ← c+1 }

  GP[r] ← c
}
return GP
```