MINISTÈRE
DE L'ÉDUCATION
NATIONALE

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

EBE MAT 1

SESSION 2019

CAPES
CONCOURS EXTERNE
ET CAFEP

Sections:

MATHÉMATIQUES
LANGUES RÉGIONALES : BRETON

PREMIÈRE ÉPREUVE D'ADMISSIBILITÉ

Durée : 5 heures

Option Mathématiques :

Calculatrice électronique de poche - y compris calculatrice programmable, alphanumérique ou à écran graphique – à fonctionnement autonome, non imprimante, autorisée conformément à la circulaire n° 99-186 du 16 novembre 1999.
L'usage de tout ouvrage de référence, de tout dictionnaire et de tout autre matériel électronique est rigoureusement interdit.

Option Informatique :

Calculatrice électronique de poche - y compris calculatrice programmable, alphanumérique ou à écran graphique – à fonctionnement autonome, non imprimante, autorisée conformément à la circulaire n° 99-186 du 16 novembre 1999.
L'usage de tout ouvrage de référence, de tout dictionnaire et de tout autre matériel électronique est rigoureusement interdit.

Les candidats doivent traiter le sujet de l'option choisie lors de l'inscription.

Si vous repérez ce qui vous semble être une erreur d'énoncé, vous devez le signaler très lisiblement sur votre copie, en proposer la correction et poursuivre l'épreuve en conséquence. De même, si cela vous conduit à formuler une ou plusieurs hypothèses, vous devez la (ou les) mentionner explicitement.

NB : Conformément au principe d'anonymat, votre copie ne doit comporter aucun signe distinctif, tel que nom, signature, origine, etc. Si le travail qui vous est demandé consiste notamment en la rédaction d'un projet ou d'une note, vous devrez impérativement vous abstenir de la signer ou de l'identifier.

Tournez la page S.V.P.

INFORMATION AUX CANDIDATS

Vous trouverez ci-après les codes nécessaires vous permettant de compléter les rubriques figurant en en-tête de votre copie.

Ces codes doivent être reportés sur chacune des copies que vous remettrez.

► Concours externe du CAPES de l'enseignement public :

- **option Mathématiques** :

Concours	Section/option	Epreuve	Matière
EBE	13OOE	101	0466

- **option Informatique** :

Concours	Section/option	Epreuve	Matière
EBE	13OOE	101	0413

- **Langue régionale Breton** :

Concours	Section/option	Epreuve	Matière
EBE	0441E	102	0466

► Concours externe du CAFEP/CAPES de l'enseignement privé :

- **option Mathématiques:**

Concours	Section/option	Epreuve	Matière
EBF	13OOE	101	0466

- **option Informatique** :

Concours	Section/option	Epreuve	Matière
EBF	13OOE	101	0413

- **Langue régionale Breton** :

Concours	Section/option	Epreuve	Matière
EBF	0441E	102	0466

OPTION MATHÉMATIQUES

Cette épreuve est constituée de deux problèmes indépendants.

Problème n° 1

Notations.
On désigne par $\mathbb{N}$ l'ensemble des entiers naturels, par $\mathbb{R}$ l'ensemble des nombres réels et par $\mathbb{C}$ l'ensemble des nombres complexes.
Pour $z \in \mathbb{C}$, on note le conjugué de z par $\overline{z}$.
Pour n un entier naturel non nul, $\mathcal{M}_n(\mathbb{C})$ désigne l'ensemble des matrices à n lignes et n colonnes, à coefficients complexes. L'ensemble des matrices inversibles pour la multiplication matricielle de $\mathcal{M}_n(\mathbb{C})$ est noté $GL_n(\mathbb{C})$.

Partie A : rotations et translations du plan

On se place dans un plan euclidien orienté $\mathcal{P}$, muni d'un repère orthonormé direct.

Notations.
Soit θ un nombre réel non congru à 0 modulo 2π et Ω un point de $\mathcal{P}$. La rotation de centre Ω et d'angle θ est notée $r_{\Omega,\theta}$.
Soit $\overrightarrow{u}$ un vecteur de $\mathcal{P}$. La translation de vecteur $\overrightarrow{u}$ est notée $t_{\overrightarrow{u}}$.

I. Question de cours. Soient θ un nombre réel non congru à 0 modulo 2π, Ω un point de $\mathcal{P}$ et $\overrightarrow{u}$ un vecteur de $\mathcal{P}$. L'affixe de Ω est notée ω et l'affixe de $\overrightarrow{u}$ est notée $z_{\overrightarrow{u}}$. Soit M un point de $\mathcal{P}$, d'affixe z. Déterminer l'affixe z' de l'image de M par $t_{\overrightarrow{u}}$. Déterminer l'affixe z'' de l'image de M par $r_{\Omega,\theta}$.

II. Soient a un nombre complexe de module 1 et b un nombre complexe. On considère l'application f de $\mathcal{P}$ dans lui-même qui a tout point d'affixe z associe le point d'affixe $az + b$.

1. Montrer que si $a = 1$, alors f est une translation dont on précisera le vecteur.

2. On suppose dans cette question que $a \neq 1$.

a. Montrer que f possède un unique point fixe Ω dont on précisera l'affixe ω.

b. Montrer que l'image par f du point M d'affixe z est le point d'affixe

$$a(z - \omega) + \omega.$$

c. Montrer que f est une rotation dont on précisera le centre et l'angle.

III. Soient a_1 et a_2 deux nombres complexes de module 1 et b_1 et b_2 deux nombres complexes. On considère l'application f_1, respectivement f_2, de $\mathcal{P}$ dans lui-même, envoyant le point d'affixe z sur le point d'affixe $a_1z + b_1$, respectivement $a_2z + b_2$.

1. Soit $f = f_1 \circ f_2$. Pour tout point M d'affixe z, calculer l'affixe de $f(M)$.

2. Montrer que f est une translation ou une rotation.

IV. Soient r_1 la rotation de centre d'affixe 1 et d'angle $\dfrac{\pi}{2}$ et r_2 la rotation de centre d'affixe 0 et d'angle $-\dfrac{\pi}{2}$. Déterminer la nature et les éléments caractéristiques de $r_1 \circ r_2$ et $r_2 \circ r_1$.

V. On considère l'ensemble G formé des rotations de $\mathcal{P}$ et des translations de $\mathcal{P}$. Montrer que G est un groupe pour une loi que l'on précisera.

Partie B : une construction géométrique

On se place de nouveau dans le plan euclidien orienté $\mathcal{P}$. On a montré dans la partie précédente que, sous certaines conditions, la composée de deux rotations est une rotation. On cherche ici à construire le centre de cette rotation.

Notations.
Soit $\mathcal{D}$ une droite de $\mathcal{P}$. La symétrie orthogonale d'axe $\mathcal{D}$ est notée $s_{\mathcal{D}}$.
Si $\overrightarrow{u}$ et $\overrightarrow{v}$ sont deux vecteurs non nuls de $\mathcal{P}$, on note $(\overrightarrow{u},\overrightarrow{v})$ l'angle orienté de $\overrightarrow{u}$ et $\overrightarrow{v}$.

VI. Soient $\mathcal{D}_1$ et $\mathcal{D}_2$ deux droites du plan, sécantes en un point Ω. On désigne par $\overrightarrow{u_1}$ et $\overrightarrow{u_2}$ des vecteurs directeurs de $\mathcal{D}_1$ et $\mathcal{D}_2$, respectivement. On considère l'application $f = s_{\mathcal{D}_2} \circ s_{\mathcal{D}_1}$.

1. Montrer que Ω est un point fixe de f.

2. Soit M un point de $\mathcal{P}$ distinct de Ω. Soient $M' = s_{\mathcal{D}_1}(M)$ et $M'' = s_{\mathcal{D}_2}(M')$. Montrer que les angles $(\overrightarrow{\Omega M}, \overrightarrow{u_1})$ et $(\overrightarrow{u_1}, \overrightarrow{\Omega M'})$ sont égaux. On montrerait de même que les angles $(\overrightarrow{\Omega M'}, \overrightarrow{u_2})$ et $(\overrightarrow{u_2}, \overrightarrow{\Omega M''})$ sont égaux.

3. Montrer que $(\overrightarrow{\Omega M}, \overrightarrow{\Omega M''}) \equiv 2(\overrightarrow{u_1}, \overrightarrow{u_2})[2\pi]$.

4. Montrer que $\Omega M = \Omega M' = \Omega M''$.

5. Montrer que f est une rotation dont on précisera le centre et l'angle.

VII. Soient r_1 et r_2 deux rotations, de centres respectifs Ω_1 et Ω_2 et d'angles respectifs θ_1 et θ_2. On suppose $\Omega_1 \neq \Omega_2$.

1. Déterminer deux droites $\mathcal{D}_1$ et $\mathcal{D}_2$ telles que $r_1 = s_{\mathcal{D}_1} \circ s_{(\Omega_1\Omega_2)}$ et $r_2 = s_{(\Omega_1\Omega_2)} \circ s_{\mathcal{D}_2}$.

2. Montrer que $r_1 \circ r_2 = s_{\mathcal{D}_1} \circ s_{\mathcal{D}_2}$.

3. On suppose $\mathcal{D}_1$ et $\mathcal{D}_2$ sécantes en un point Ω. Montrer qu'alors $r_1 \circ r_2$ est une rotation dont on précisera le centre et l'angle.

4. Donner une construction à la règle et au compas du centre de la rotation $r_1 \circ r_2$ lorsque r_1 est la rotation de centre d'affixe i et d'angle $\dfrac{\pi}{2}$ et r_2 est la rotation de centre O et d'angle $\dfrac{\pi}{3}$.

5. Que se passe t-il si $\mathcal{D}_1$ et $\mathcal{D}_2$ sont parallèles ?

Partie C : structure des quaternions

Soient a et b deux nombres complexes. On note $M(a,b)$ la matrice complexe suivante :

$$M(a,b) = \begin{pmatrix} a & -b \\ \overline{b} & \overline{a} \end{pmatrix}.$$

Une matrice $M \in \mathcal{M}_2(\mathbb{C})$ de la forme $M(a,b)$ est appelée un quaternion. On considère en particulier les quaternions suivants :

$$E = M(1,0), \qquad I = M(i,0), \qquad J = M(0,1), \qquad K = M(0,i).$$

On veillera à ne pas confondre la matrice $I = M(i,0)$ avec la matrice identité d'ordre 2, $I_2 = E$.

On note $\mathbb{H} = \{M(a,b) \mid (a,b) \in \mathbb{C}^2\}$.

VIII. **1.** Donner sans justification une base du $\mathbb{C}$-espace vectoriel $\mathcal{M}_2(\mathbb{C})$ puis une base du $\mathbb{R}$-espace vectoriel $\mathcal{M}_2(\mathbb{C})$.

2. Montrer que $\mathbb{H}$ est un sous-espace vectoriel du $\mathbb{R}$-espace vectoriel $\mathcal{M}_2(\mathbb{C})$, dont une base est (E, I, J, K).

En conséquence, tout quaternion q s'écrit de manière unique $q = xE+yI+zJ+tK$, avec $x, y, z, t \in \mathbb{R}$.

3. Pour a, b, a', b' des nombres complexes, calculer $M(a,b)M(a',b')$. En déduire que $\mathbb{H}$ est stable par la multiplication matricielle.

IX. **1.** Calculer les produits deux à deux des matrices E, I, J et K. On présentera les résultats dans un tableau à double entrée.

2. La multiplication dans $\mathbb{H}$ est-elle commutative ?

X. Montrer que tout quaternion $q = M(a,b)$ avec $(a,b) \in \mathbb{C}^2 \setminus \{(0,0)\}$, est un élément de $GL_2(\mathbb{C})$ dont l'inverse q^{-1} est un quaternion.

XI. Montrer que $\{q \in \mathbb{H} \mid \forall r \in \mathbb{H}, qr = rq\} = \{xE \mid x \in \mathbb{R}\}$.

Partie D : conjugué, parties réelle et imaginaire d'un quaternion

Soit $q = xE + yI + zJ + tK \in \mathbb{H}$, avec $x, y, z, t \in \mathbb{R}$. On définit le quaternion conjugué de q, noté q^*, par :

$$q^* = xE - yI - zJ - tK.$$

On définit la partie réelle de q, notée $\mathcal{R}e(q)$, par $\mathcal{R}e(q) = xE$.
On définit la partie imaginaire de q, notée $\mathcal{I}m(q)$, par $\mathcal{I}m(q) = yI + zJ + tK$.
On définit l'ensemble des quaternions purs, noté $\mathbb{H}_{pur}$, par $\mathbb{H}_{pur} = \{q \in \mathbb{H} \mid \mathcal{R}e(q) = 0\}$.

XII. **1.** Soit q un quaternion. Montrer que q^* est la transposée de la matrice obtenue en conjugant tous les coefficients de q.

2. En déduire que, pour tous quaternions q, r, $(qr)^* = r^*q^*$.

XIII. Pour tout quaternion q, on pose $N(q) = qq^*$.

1. Montrer que, pour tout quaternion $q = xE + yI + zJ + tK$, avec $x, y, z, t \in \mathbb{R}$, $N(q) = (x^2 + y^2 + z^2 + t^2)E$.

2. Montrer que, pour tous quaternions q, r, $N(qr) = N(q)N(r)$.

Partie E : norme sur $\mathbb{H}$

On admet qu'on définit une norme euclidienne sur $\mathbb{H}$ de la façon suivante :

$$\begin{cases} \mathbb{H} & \longrightarrow & \mathbb{R} \\ q = xE + yI + zJ + tK & \mapsto & ||q|| = \sqrt{x^2 + y^2 + z^2 + t^2} \end{cases}$$

XIV. Quel est le produit scalaire associé à cette norme euclidienne ?

Tournez la page S.V.P.

XV. **1.** Montrer que, pour tout quaternion q, $N(q) = ||q||^2 E$.

2. En déduire que, pour tous quaternions q, r, $||qr|| = ||q|| \times ||r||$.

3. En déduire que pour tout quaternion non nul q, $||q^{-1}|| = \dfrac{1}{||q||}$.

XVI. On considère l'application suivante :

$$\psi : \left\{ \begin{array}{ccc} \mathbb{R}^3 & \longrightarrow & \mathbb{H}_{pur} \\ \overrightarrow{q} = (y,z,t) & \mapsto & q = yI + zJ + tK. \end{array} \right.$$

Le quaternion pur $\psi(\overrightarrow{q})$ est appelé quaternion pur associé au vecteur $\overrightarrow{q}$. L'espace $\mathbb{R}^3$ est muni de sa structure euclidienne canonique et est supposé orienté. Son produit scalaire est noté $\langle \cdot \mid \cdot \rangle$. De plus, $\mathbb{H}_{pur}$ est muni de la structure euclidienne induite par celle de $\mathbb{H}$.

1. Montrer que ψ est une isométrie, c'est-à-dire que pour tout $\overrightarrow{q} \in \mathbb{R}^3$,

$$||\psi(\overrightarrow{q})|| = ||\overrightarrow{q}||.$$

2. Soient $q_1, q_2 \in \mathbb{H}_{pur}$, respectivement associés aux vecteurs $\overrightarrow{q_1}$ et $\overrightarrow{q_2}$. Montrer que $\mathcal{R}e(q_1 q_2) = -\langle \overrightarrow{q_1} \mid \overrightarrow{q_2} \rangle E$ et que $\mathcal{I}m(q_1 q_2) = \psi(\overrightarrow{q_1} \wedge \overrightarrow{q_2})$, où $\overrightarrow{q_1} \wedge \overrightarrow{q_2}$ désigne le produit vectoriel des vecteurs $\overrightarrow{q_1}$ et $\overrightarrow{q_2}$.

3. Soit $q \in \mathbb{H}_{pur}$. Calculer $\mathcal{R}e(q^2)$ et $\mathcal{I}m(q^2)$. En déduire q^2.

4. Soient $(a,b,c,d) \in \mathbb{R}^4$ et $q \in \mathbb{H}_{pur}$. Calculer $(aE + bq)(cE + dq)$.

5. Soient q_1 et q_2 deux quaternions purs, respectivement associés aux vecteurs $\overrightarrow{q_1}$ et $\overrightarrow{q_2}$. Montrer que $\langle \overrightarrow{q_1} \mid \overrightarrow{q_2} \rangle = 0$ si et seulement si $q_1 q_2 + q_2 q_1 = 0$.

Partie F : quaternions unitaires et rotations vectorielles

On note $U = \{q \in \mathbb{H} \mid N(q) = E\}$. Les éléments de U sont appelés quaternions unitaires.

XVII. Montrer que U est un sous-groupe de $GL_2(\mathbb{C})$.

XVIII. Soit $p \in U$.

1. Montrer qu'il existe un nombre réel θ et un quaternion $u \in U \cap \mathbb{H}_{pur}$ tel que

$$p = \cos(\theta)E + \sin(\theta)u.$$

2. Vérifier que $p^{-1} = p^* = \cos(\theta)E - \sin(\theta)u$.

XIX. Soit $p \in U$. On définit l'application suivante :

$$r_p : \left\{ \begin{array}{ccc} \mathbb{H} & \longrightarrow & \mathbb{H} \\ q & \mapsto & pqp^{-1}. \end{array} \right.$$

1. Montrer que r_p est une application linéaire.

2. Montrer que pour tout $q \in \mathbb{H}$, $||r_p(q)|| = ||q||$.

3. Soient p_1 et p_2 deux éléments de U. Montrer que $r_{p_1} \circ r_{p_2} = r_{p_1 p_2}$. En déduire que pour tout $p \in U$, r_p est une bijection d'inverse $r_{p^{-1}}$.

4. Montrer que r_p est égale à l'identité de $\mathbb{H}$ si et seulement si $p = E$ ou $p = -E$.

5. Soient p_1 et p_2 deux quaternions unitaires. Déduire de la question précédente que $r_{p_1} = r_{p_2}$ si et seulement si $p_1 = p_2$ ou $p_1 = -p_2$.

XX. On suppose maintenant que p est un quaternion unitaire différent de E et de $-E$. D'après la question XVIII. 1., le quaternion p s'écrit sous la forme $p = \cos(\theta)E + \sin(\theta)u$, où θ est un nombre réel u est un quaternion pur unitaire. On associe à u le vecteur $\overrightarrow{u}$ par l'application ψ définie dans la question ψ définie dans la question XVI.
Soit $\overrightarrow{v}$ un vecteur unitaire de $\mathbb{R}^3$ orthogonal à $\overrightarrow{u}$. On pose $\overrightarrow{w} = \overrightarrow{u} \wedge \overrightarrow{v}$. On note v et w les quaternions purs associés aux vecteurs $\overrightarrow{v}$ et $\overrightarrow{w}$.

1. Que peut-on dire de la famille $(\overrightarrow{u}, \overrightarrow{v}, \overrightarrow{w})$?

2. Montrer que $uv = -vu = w$, $uw = -wu = -v$, $u^2 = -E$ et que $u^3 = -u$.

3. Calculer $r_p(u)$, $r_p(v)$ et $r_p(w)$.

4. Montrer qu'il existe une rotation vectorielle de $\mathbb{R}^3$ notée R, dont on précisera l'axe et l'angle, telle que pour tout $q \in \mathbb{H}_{pur}$, si $q = \psi(\overrightarrow{q})$, alors $r_p(q) = \psi(R(\overrightarrow{q}))$.

XXI. Soit R une rotation vectorielle de l'espace euclidien $\mathbb{R}^3$, d'axe la droite D dirigée par un vecteur unitaire $\overrightarrow{d}$ et d'angle ϕ. Montrer qu'il existe $p \in U$ tel que pour tout $q \in \mathbb{H}_{pur}$, si $q = \psi(\overrightarrow{q})$, alors $r_p(q) = \psi(R(\overrightarrow{q}))$.

XXII. Application. Soient R_1 la rotation vectorielle de $\mathbb{R}^3$ d'angle $\dfrac{2\pi}{3}$ et d'axe engendré par $(1, -1, -1)$ et R_2 la rotation vectorielle de $\mathbb{R}^3$ d'angle π et d'axe engendrée par $(0, 1, 0)$. Montrer que $R_2 \circ R_1$ et $R_1 \circ R_2$ sont des rotations dont on précisera les axes et les angles.

Tournez la page S.V.P.

Problème n° 2

Notations.
On désigne par $\mathbb{N}$ l'ensemble des entiers naturels, par $\mathbb{N}^*$ l'ensemble des entiers naturels non nuls et par $\mathbb{R}$ l'ensemble des nombres réels.
Soit $(\Omega, \mathcal{B}, \mathbb{P})$ un espace probabilisé. Si A et B sont deux événements de Ω avec B de probabilité non nulle, la probabilité conditionnelle de A sachant que B est réalisé est notée $\mathbb{P}_B(A)$.
Soient k et n des entiers naturels, avec $0 \leqslant k \leqslant n$. Le coefficient binomial donnant le nombre de parties à k éléments est noté $\binom{n}{k}$.

On utilisera la convention $0^0 = 1$ dans tout le problème.

Partie A : quelques études de séries

I. **1.** Montrer que, pour tout entier naturel n et tout nombre réel x différent de 1,

$$\sum_{k=0}^{n} x^k = \frac{1 - x^{n+1}}{1 - x}.$$

2. En déduire, pour tout entier naturel n non nul et tout nombre réel x différent de 1, une expression de $\displaystyle\sum_{k=1}^{n} kx^{k-1}$.

3. Soit $x \in]-1\,;1[$. En déduire la convergence de la série $\displaystyle\sum_{n\geq 1} nx^{n-1}$ et donner la valeur de sa somme.

II. Soit k un entier naturel. On considère la série entière

$$S_k(x) = \sum_{n=k}^{+\infty} \binom{n}{k} x^{n-k}.$$

1. Calculer le rayon de convergence de $S_k(x)$.

2. Montrer que S_k est dérivable sur $]-1\,;1[$ et que, pour tout $x \in]-1\,;1[$,

$$S_k'(x) = (k+1)S_{k+1}(x).$$

3. Montrer par récurrence que, pour tout $k \in \mathbb{N}$ et pour tout $x \in]-1\,;1[$,

$$S_k(x) = \frac{1}{(1-x)^{k+1}}.$$

4. Soit $x \in]-1\,;1[$. Justifier la convergence de la série $\displaystyle\sum_{n\geq 1} n^2 x^{n-1}$ et montrer que

$$\sum_{n=1}^{+\infty} n^2 x^{n-1} = \frac{x+1}{(1-x)^3}.$$

Indication : on pourra écrire n^2 en fonction de $\binom{n}{1}$ et de $\binom{n}{2}$.

III. Application : soit $(\Omega, \mathcal{A}, P)$ un espace probabilisé. Soit p un réel de $]0\,;1[$. Soit X une variable aléatoire discrète suivant la loi géométrique de paramètre p, c'est-à-dire une variable aléatoire définie sur Ω, telle que

$$X(\Omega) = \mathbb{N}^* \text{ et } \forall k \in \mathbb{N}^*,\ \mathbb{P}(X = k) = p(1-p)^{k-1}.$$

1. Montrer que X admet une espérance et la calculer.

2. Montrer que X^2 admet une espérance et la calculer.

3. Montrer que X admet une variance et la calculer.

Partie B : étude d'une séance de tir à l'arc

On considère deux archers A_1 et A_2 qui tirent chacun sur une cible de manière indépendante. L'archer A_1 (respectivement A_2) touche sa cible avec une probabilité p_1 (respectivement p_2) strictement comprise entre 0 et 1. On suppose de plus que les tirs des joueurs sont indépendants les uns des autres. On appelle X_1 (respectivement X_2) la variable aléatoire donnant le nombre de tirs nécessaires à l'archer A_1 (respectivement A_2) pour qu'il touche sa cible pour la première fois. On note $q_1 = 1 - p_1$ et $q_2 = 1 - p_2$.

IV. Déterminer les valeurs possibles prises par X_1.

V. On introduit, pour tout entier naturel non nul i, l'événement E_i : « le joueur A_1 touche la cible à son i-ème tir ».
Exprimer, pour tout $k \in \mathbb{N}^*$, l'événement $(X_1 = k)$ à l'aide des événements E_i, $i \in \mathbb{N}^*$.

VI. En déduire la loi de X_1.

VII. **1.** Pour tout entier naturel non nul k, calculer $\mathbb{P}(X_1 > k)$.

2. Montrer que

$$\forall (m,n) \in \mathbb{N}^* \times \mathbb{N}^*,\ \mathbb{P}_{(X_1>m)}(X_1 > n+m) = \mathbb{P}(X_1 > n).$$

VIII. Calculer $\mathbb{P}(X_1 = X_2)$.

IX. Calculer $\mathbb{P}(X_1 > X_2)$.

X. Que vaut $\mathbb{P}(X_2 > X_1)$?

XI. On réalise à présent une deuxième expérience avec les deux archers A_1 et A_2 de la manière suivante : l'archer A_1 tire jusqu'à ce qu'il touche sa cible. On appelle X_1 la variable aléatoire donnant le nombre de tirs effectués par le joueur A_1 pour qu'il touche sa cible pour la première fois. Ensuite, si X_1 prend la valeur n, l'archer A_2 effectue n tirs en direction de sa cible dans les mêmes conditions que la première expérience.
On définit alors la variable aléatoire G égale au nombre de fois où la cible a été touchée par l'archer A_2. On suppose dans cette partie que $p_1 = p_2$ et on note

$$p = p_1 = p_2, \qquad q = 1 - p = 1 - p_1 = 1 - p_2.$$

1. Soient $n \in \mathbb{N}^*$ et $k \in \mathbb{N}$. Déterminer la probabilité conditionnelle $P_{(X_1=n)}(G = k)$. On distinguera les cas $k > n$ et $k \leq n$.

2. Montrer que, pour tout $k \in \mathbb{N}$, $\mathbb{P}(G = k) = q^{k-1}p^{k+1}\displaystyle\sum_{n=k}^{+\infty} \binom{n}{k} q^{2n-2k}$.

Tournez la page S.V.P.

3. En utilisant la partie **A.**, montrer que, pour tout $k \in \mathbb{N}$,

$$\mathbb{P}(G = k) = \left(\frac{q}{1+q}\right)^{k-1} \times \frac{1}{(1+q)^2}.$$

4. Montrer que G admet une espérance et que celle-ci vaut 1. Interpréter ce résultat.

Partie C : étude d'une variable discrète sans mémoire

Soit Y une variable aléatoire discrète, à valeurs dans $\mathbb{N}$ telle que pour tout entier naturel n, $\mathbb{P}(Y \geqslant n) > 0$.
On suppose également que Y est sans mémoire c'est-à-dire qu'elle vérifie :

$$\forall (m,n) \in \mathbb{N} \times \mathbb{N},\ \mathbb{P}_{(Y \geqslant m)}(Y \geqslant n+m) = \mathbb{P}(Y \geqslant n).$$

On pose $\mathbb{P}(Y = 0) = p$ et $q = 1 - p$.

XII. Montrer que $\mathbb{P}(Y \geqslant 1) = q$. En déduire que $0 < q \leqslant 1$.

XIII. Montrer que pour tout couple (m,n) d'entiers naturels,

$$\mathbb{P}(Y \geqslant n+m) = \mathbb{P}(Y \geqslant m)\mathbb{P}(Y \geqslant n).$$

XIV. Pour tout entier naturel n, on pose $u_n = \mathbb{P}(Y \geqslant n)$.

1. Montrer que la suite (u_n) est géométrique et préciser sa raison.

2. Pour tout entier naturel n, exprimer $\mathbb{P}(Y \geqslant n)$ en fonction de n et de q.

3. Montrer que pour tout entier naturel n, $\mathbb{P}(Y = n) = \mathbb{P}(Y \geqslant n) - \mathbb{P}(Y \geqslant n+1)$.

4. En déduire que, pour tout $n \in \mathbb{N}$, $\mathbb{P}(Y = n) = q^n p$.

5. En déduire que q est différent de 1.

XV. Reconnaître la loi suivie par la variable aléatoire $Y + 1$.

XVI. Conclure que Y est sans mémoire si et seulement si $Y + 1$ est une variable aléatoire de loi géométrique de paramètre $p \in]0\,;1[$.

Partie D : taux de panne d'une variable discrète

Soit Z une variable aléatoire à valeurs dans $\mathbb{N}$ telle que, pour tout entier naturel n,

$$\mathbb{P}(Z \geqslant n) > 0.$$

Soit $n \in \mathbb{N}$. On appelle taux de panne de Z à l'instant n, le réel noté λ_n défini par

$$\lambda_n = \mathbb{P}_{(Z \geqslant n)}(Z = n).$$

XVII. **1.** Montrer que, pour tout entier naturel n,

$$1 - \lambda_n = \frac{\mathbb{P}(Z \geqslant n+1)}{\mathbb{P}(Z \geqslant n)}.$$

2. Vérifier alors que, pour tout entier naturel n, on a $0 \leqslant \lambda_n < 1$.

3. Montrer que, pour tout entier naturel n non nul,

$$\mathbb{P}(Z \geqslant n) = \prod_{k=0}^{n-1}(1-\lambda_k).$$

XVIII. **1.** Montrer que, pour tout entier naturel n non nul,

$$\sum_{k=0}^{n-1}\mathbb{P}(Z=k) = 1 - \mathbb{P}(Z \geqslant n).$$

2. En déduire que $\lim\limits_{n\to\infty}\mathbb{P}(Z \geqslant n)$ existe et vaut 0.

3. Quelle est la nature de la série $\sum\limits_{n\geq 0}\ln(1-\lambda_n)$?

4. Que dire alors de la nature de la série $\sum\limits_{n\geq 0}\lambda_n$?

XIX. On suppose maintenant qu'il existe un nombre réel c tel que pour tout $n \in \mathbb{N}$, $\lambda_n = c$. Ce réel est appelé taux de panne de Z.

1. Montrer que $0 \leqslant c < 1$.

2. Pour tout entier naturel n, exprimer $\mathbb{P}(Z \geqslant n)$ en fonction de c et de n.

3. Montrer que c est non nul.

4. En déduire une caractérisation des variables aléatoires ayant un taux de panne constant.

OPTION INFORMATIQUE

Cette épreuve est constituée de deux problèmes indépendants.

On rappelle que tous les programmes demandés doivent être rédigés dans le langage Python.

Problème 1 : base 3 équilibrée

Notation. $\mathbb{N}^*$ désigne l'ensemble des entiers naturels non nuls.

En informatique, on utilise traditionnellement la base 2 pour représenter les entiers. Ce choix est dicté par la technologie utilisée par les ordinateurs actuels. Dans les années 60, d'autres technologies avaient été envisagées, dont l'utilisation d'une base de numération appelée base 3 équilibrée.

En base 3 classique, on écrit les nombres à l'aide de trois chiffres : 0, 1 et 2 : par exemple, $23 = \overline{212}^3 = 2 \times 3^2 + 1 \times 3 + 2$, $46 = \overline{1201}^3 = 1 \times 3^3 + 2 \times 3^2 + 0 \times 3 + 1$.

En base 3 équilibrée, les « chiffres » utilisés sont 0, 1 et -1 :

$23 = \overline{10(-1)(-1)}^e = 1 \times 3^3 + 0 \times 3^2 - 1 \times 3 - 1$, $46 = \overline{1(-1)(-1)01}^e = 1 \times 3^4 - 1 \times 3^3 - 1 \times 3^2 + 0 \times 3 + 1$.

Dans toute la suite, on s'intéresse à l'écriture en base 3 équilibrée des entiers naturels **non nuls**. On note $C = \{-1, 0, 1\}$ l'ensemble des chiffres utilisés dans l'écriture en base 3 équilibrée.

L'écriture en base 3 équilibrée d'un entier naturel **non nul** sera notée comme ci-dessus, par la suite des chiffres surmontée d'un trait avec un exposant e. Si a_0, a_1, …, a_{p-1} sont dans C, on a donc :

$$\overline{a_0 a_1 \dots a_{p-1}}^e = \sum_{k=0}^{p-1} a_k 3^{p-1-k}.$$

L'écriture en base 3 équilibrée d'un entier naturel non nul, lue de gauche à droite, commence toujours par le chiffre 1.

Partie I – généralités

Question 1.

1.a Donner la valeur de $\overline{1(-1)0(-1)}^e$, de $\overline{1111}^e$ et de $\overline{1(-1)(-1)(-1)(-1)}^e$.

1.b Écrire en base 3 équilibrée les entiers de 1 à 9.

Question 2. Soit k un entier naturel.

Déterminer la valeur de $A_k = \overline{1\underbrace{11\dots1}_{k \text{ chiffres}}}^e$ et de $B_k = \overline{1\underbrace{(-1)(-1)\dots(-1)}_{k \text{ chiffres}}}^e$.

(On aura bien noté que ces deux écritures possèdent $k+1$ chiffres au total.)

Question 3. On représente l'écriture en base 3 équilibrée $\overline{a_0 \dots a_{p-1}}^e$ par la liste Python $[a_{p-1}, a_{p-2}, \dots, a_0]$ (attention, les chiffres sont écrits dans la liste en lisant de droite à gauche l'écriture en base 3 équilibrée, le dernier d'entre eux est donc toujours le chiffre 1).

Écrire une fonction `valeur(L)` qui prend en argument une liste de chiffres et renvoie l'entier naturel non nul dont c'est une écriture en base 3 équilibrée.

Par exemple `valeur([1,0,-1,-1,1])` renvoie l'entier 46.

Partie II – existence et unicité de l'écriture en base 3 équilibrée

Question 4. On veut montrer que tout entier naturel non nul n admet une écriture en base 3 équilibrée.

Soit $n \in \mathbb{N}^*$. On note q et r le quotient et le reste dans la division euclidienne de n par 3 : $n = 3q + r$ et $r \in \{0, 1, 2\}$.

4.a On suppose que q est non nul et admet une écriture en base 3 équilibrée $q = \overline{a_0 \dots a_{p-1}}^e$. Déterminer une écriture en base 3 équilibrée de n dans le cas où $r = 0$ ou $r = 1$.

4.b On suppose que q est non nul et que $q+1$ admet une écriture en base 3 équilibrée $q + 1 = \overline{b_0 \dots b_{p-1}}^e$. Déterminer une écriture en base 3 équilibrée de n dans le cas où $r = 2$.

4.c Montrer par récurrence que tout entier naturel non nul admet une écriture en base 3 équilibrée.

Question 5. On souhaite écrire en Python une fonction incremente(L) qui prend en argument une liste L de chiffres représentant une écriture en base 3 équilibrée d'un entier naturel non nul n et qui renvoie une liste représentant une écriture en base 3 équilibrée de $n+1$.

5.a On propose la fonction récursive incrementeR(L) suivante.

Expliquer la ligne 9. Pourquoi a-t-il fallu écrire les lignes 1 et 2 ?

```
0 def incrementeR(L):
1     if len(L)==0:
2         return [1]
3     elif L[0]==0:
4         return [1]+L[1:]
5     elif L[0]==-1:
6         return [0]+L[1:]
7     else:
8         # ici L[0]==1
9         return [-1]+incrementeR(L[1:])
```

5.b On souhaite écrire une version non récursive incremente(L) de cette fonction.

Recopier et compléter le script suivant pour répondre à cette question :

```
def incremente(L):
    p = len(L)
    M = [] # liste finale
    k = 0
    while k < p and L[k] == 1:
        M.append(-1)
        k += 1
    if k == p:
        M.append(...)
    elif L[k] == 0:
        M.append(...)
        M = M + L[k+1:]
    elif L[k] == -1:
        ...
        ...
    return M
```

Question 6. On souhaite montrer que l'écriture en base 3 équilibrée d'un entier naturel n non nul est unique.

6.a Soit $a_0, \ldots, a_{p-1}$ des chiffres de C (avec $a_0 = 1$ et $p \geqslant 1$). Quel est le reste dans la division euclidienne par 3 du nombre $\overline{a_0 a_1 \ldots a_{p-1}}^e$?

6.b En déduire que l'écriture en base 3 équilibrée d'un entier naturel n non nul est unique.

Question 7. Écrire une fonction base3e(n) qui prend en argument un entier naturel n non nul et renvoie la liste L de chiffres qui correspond à l'écriture en base 3 équilibrée de n. Par exemple base3e(46) renvoie [1,0,−1,−1,1]).

Partie III – chemins de Delannoy

On considère le plan euclidien P rapporté à un repère orthonormé.

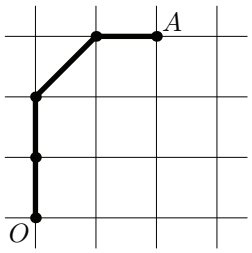
Un chemin de Delannoy du plan P est un arc suivant une ligne brisée partant de l'origine O et arrivant en un point A de coordonnées (a, b) où a et b sont deux entiers naturels, constituée d'arcs à choisir parmi trois types : un arc horizontal (associé au vecteur $(1, 0)$), un arc vertical (associé au vecteur $(0, 1)$) ou un arc diagonal (associé au vecteur $(1, 1)$). Dans le parcours d'un chemin de Delannoy, on va donc toujours vers la droite, vers le haut, ou en diagonale vers la droite et vers le haut en même temps.

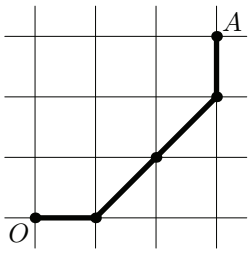
On associe à tout chemin de Delannoy un entier naturel non nul n défini par son écriture en base 3 équilibrée de la façon suivante : le premier chiffre de cette écriture est toujours égal à 1 (comme pour toute écriture en base 3 équilibrée), les chiffres suivants caractérisent les arcs consécutifs du chemin, le chiffre 1 est associé à un segment horizontal, le chiffre -1 à un arc vertical, et le chiffre 0 à un arc diagonal. Inversement à tout entier naturel non nul on peut associer le chemin de Delannoy qui est associé à son écriture en base 3 équilibrée.

On fera attention que le premier chiffre de l'écriture en base 3 équilibrée, toujours égal à 1, n'est pas pris en compte dans la construction du chemin.

Remarque : $n = 1 = \overline{1}^e$ est l'entier associé à un chemin de Delannoy de longueur nulle.

Par exemple, les entiers associés aux deux chemins de Delannoy ci-dessous sont respectivement $46 = \overline{1(-1)(-1)01}^e$ pour la figure de gauche et $107 = \overline{1100(-1)}^e$ pour la figure de droite.





Question 8.

8.a Dessiner le chemin de Delannoy associé à l'entier 2019.

8.b On note (a, b) les coordonnées entières du point d'arrivée d'un chemin de Delannoy. Écrire la fonction `arrivee(n)` qui renvoie ce couple de coordonnées quand on lui donne en argument l'entier non nul n associé au chemin de Delannoy. Par exemple `arrivee(46)` renvoie le couple `(2, 3)` et `arrivee(107)` renvoie le couple `(3, 3)`.

On pourra s'inspirer de la fonction `base3e`.

Question 9.

Soit a et b deux entiers naturels, et A le point de coordonnées (a, b). Il existe plusieurs chemins de Delannoy allant de l'origine au point A, chacun d'eux étant associé à un entier naturel. Soit $N(a, b)$ l'ensemble de ces entiers.

9.a Soit $a \in \mathbb{N}^*$. Déterminer $\min N(a, a)$ et $\max N(a, a)$.

9.b On suppose que les entiers a et b vérifient $0 < a < b$. Déterminer $\min N(a, b)$ et $\max N(a, b)$.

Problème 2 : compilation et algorithmes

Notation. Pour m et n deux entiers naturels, $[\![m, n]\!]$ désigne l'ensemble des entiers k tels que $m \leqslant k \leqslant n$.

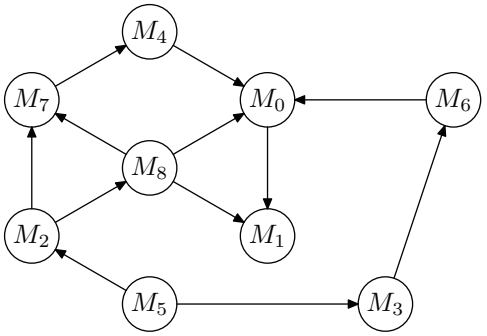
Partie A – ordre topologique sur un graphe

Lors de l'écriture d'un programme complexe, on le découpe souvent en modules distincts, mais dépendants les uns des autres. Le compilateur a besoin de connaître l'ordre dans lequel compiler ces modules. Pour ce faire, on utilise un graphe de dépendances, que nous décrivons brièvement ici.

On suppose qu'on dispose de n modules numérotés M_0, M_1, …, M_{n-1} et on représente leurs relations de dépendance à l'aide d'un graphe Γ dont les sommets sont les modules, et dont une arête allant du module M_i au module M_j indique que le module M_i doit être compilé avant le module M_j.

Par exemple, avec $n = 9$, aux contraintes du tableau de gauche ci-dessous, on peut associer le graphe Γ de droite.

le module	est compilé APRÈS les modules
M_0	M_4, M_6, M_8
M_1	M_0, M_8
M_2	M_5
M_3	M_5
M_4	M_7
M_5	–
M_6	M_3
M_7	M_2, M_8
M_8	M_2



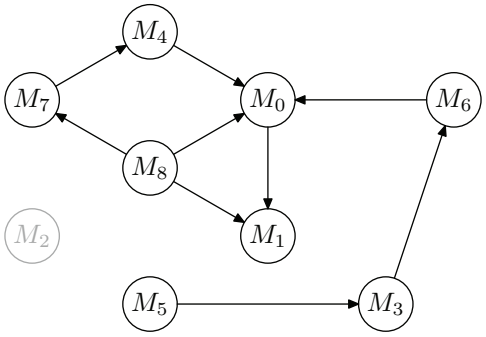
Tournez la page S.V.P.

On aura besoin de travailler sur des sous-graphes G du graphe Γ. C'est pourquoi, pour toute la suite, on suppose fixé l'entier n (qu'on pourra supposer supérieur à 2), et les graphes considérés n'utiliseront que quelques-uns des n sommets M_0, M_1, …, M_{n-1}. Pour simplifier la rédaction, on pourra utiliser la notation suivante : si s est un sommet d'un graphe G, on note $[s]$ son numéro, de sorte que $s = M_{[s]}$.

On représente un graphe G dont les sommets sont pris parmi les n sommets listés ci-dessus par une liste G=[App,Succ,p] où App est une liste de n valeurs booléennes (c'est-à-dire True ou False) ; Succ est une liste de n listes ; p est un entier. Plus précisément :

- pour $i \in [\![0, n-1]\!]$, App[i] est True si et seulement si le module M_i est présent dans le graphe ;
- p est le nombre de sommets du graphe considéré, donc le nombre d'occurrences de True dans la liste App ;
- pour $i \in [\![0, n-1]\!]$, Succ[i] est la liste vide si M_i n'est pas présent dans le graphe, et sinon Succ[i] est la liste (éventuellement vide) des numéros j des sommets M_j tels qu'il y a un arc de M_i vers M_j.

Par exemple, le graphe G de la figure suivante (obtenu en supprimant dans Γ le sommet M_2) :



est représenté par la liste G=[App,Succ,p] où

- App = [True,True,False,True,True,True,True,True,True] ;
- Succ = [[1], [], [], [6], [0], [3], [0], [4], [0, 1, 7]] ;
- p=8.

Dans toute la suite, on suppose que la valeur n, qui est fixée, est stockée dans une variable globale n. Les variables i et j désignent des numéros de sommets, donc des entiers de l'intervalle $[\![0, n-1]\!]$.

Question 10.

Écrire une fonction mem(i,G) qui prend en argument un entier i et la représentation G d'un graphe (telle qu'elle est décrite ci-dessus) et renvoie le booléen indiquant si le sommet de numéro i est dans le graphe.

Question 11.

Écrire une fonction pred(i,G) qui renvoie la liste, éventuellement vide, des numéros j des sommets du graphe représenté par G tels que l'arc qui va de M_j à M_i soit présent dans le graphe.

Par exemple, pour le graphe de la figure ci-dessus, pred(0,G) renvoie la liste [4, 6, 8].

Question 12.

Écrire une fonction sansSuccesseur(G) qui renvoie le numéro i d'un sommet M_i du graphe qui n'a pas de successeur, s'il en existe, ou qui renvoie -1 s'il n'y a pas de tel sommet. Dans l'exemple du graphe ci-dessus, sansSuccesseur(G) renvoie 1.

Question 13.

Compléter la fonction suivante suppr(i,G) qui renvoie **un nouveau graphe** H obtenu à partir de G en supprimant le sommet de numéro i et toutes les flèches qui lui sont reliées.

```
def suppr(i,G):
    H = copy.deepcopy(G)
    H[2] = H[2] - 1
    ...
      .
      .
      .
    ...
    return H
```

L'appel H = copy.deepcopy(G) permet d'instancier la variable H avec une copie de G.

Étant donné un graphe de dépendances G possédant p sommets (avec $p \leqslant n$), un ordre topologique sur le graphe est la donnée d'une renumérotation des sommets du graphe qui respecte les arcs du graphe, c'est-à-dire une fonction N bijective, définie sur l'ensemble des numéros des sommets présents dans le graphe G, à valeurs dans $[\![0, p-1]\!]$ telle que s'il existe un arc du sommet M_i vers le sommet M_j on a $N(i) < N(j)$.

Par exemple, pour le graphe de la dernière figure ci-dessus, il existe un ordre topologique, défini par $N(0) = 6$, $N(1) = 7$, $N(3) = 1$, $N(4) = 5$, $N(5) = 0$, $N(6) = 2$, $N(7) = 4$ et $N(8) = 3$.

Un chemin dans un graphe est une suite d'au moins deux sommets reliés par des arcs consécutifs. Un cycle est un chemin qui retourne au sommet de départ.

Question 14.

Dans cette question, on considère un graphe orienté G possédant p sommets (avec $p \leqslant n$) qui sont pris parmi M_0, M_1, …, M_{n-1}. On suppose que chaque sommet admet au moins un successeur.

14.a On effectue une promenade dans le graphe : on choisit un sommet s_0, puis un successeur s_1 de s_0, puis un successeur s_2 de s_1, etc. On peut ainsi obtenir une suite s_0, s_1, …, s_n. Montrer qu'il existe deux indices i et j tels que $0 \leqslant i < j \leqslant n$ et $s_i = s_j$.

14.b Justifier qu'il existe au moins un cycle dans ce graphe.

14.c Montrer qu'il est impossible de trouver un ordre topologique sur ce graphe.

Question 15.

Soit G un graphe ayant p sommets (avec $p \leqslant n$) parmi M_0, …, M_{n-1}. On suppose que G admet au moins un sommet s sans successeur. On appelle H le graphe obtenu à partir de G en supprimant le sommet s et tous les arcs reliés à s. Montrer que si H possède un ordre topologique N_H à valeurs dans $[\![0, p-2]\!]$, on peut l'étendre à un ordre topologique N sur G en posant $N([s]) = p - 1$ et $N([s']) = N_H([s'])$ pour tous les sommets s' de H.

Question 16.

On peut déduire de cette étude un algorithme de détermination d'un ordre topologique N sur un graphe G, quand il en existe. Un tel ordre topologique est représenté par une liste L à n éléments : si le sommet M_i n'est pas dans le graphe G, on a L[i] $= -1$; si M_i est dans G, L[i] contient la valeur $N(i)$.

16.a Recopier et compléter la fonction suivante pour répondre à la question : ordreTopologique(G) renvoie None s'il n'existe pas d'ordre topologique sur le graphe G et une liste L qui représente un ordre topologique s'il en existe un. La fonction parcoursReussi peut procéder à des appels récursifs.

```
def ordreTopologique(G):
  n = len(G[0])
    L = [-1] * n

    def parcoursReussi(G):
        p = G[2]
        if p != 0:
            s = sansSuccesseur(G)
            if s == -1:
                return ...
            else:
                ...
                ...
                return ...
        else:
            return ...


    b = parcoursReussi(G)
    if b:
        return L
    else:
        return None
```

16.b Prouver que l'algorithme termine toujours.

Tournez la page S.V.P.

Partie B – allocation de registres

Quand le code est en phase finale de compilation, on peut supposer, pour simplifier, qu'il se présente comme une succession d'affectations de variables ou d'utilisation de ces variables à l'aide de fonctions simples (opérations arithmétiques ou logiques). L'accès à la mémoire étant plus coûteux en temps que le calcul lui-même, il est préférable de conserver le plus possible de résultats intermédiaires dans les registres du processeur plutôt que dans la mémoire vive. Mais le processeur n'a qu'un nombre limité de registres : il s'agit donc de savoir si tous les calculs peuvent être menés en se confinant à cet espace limité.

On se limite ici à un modèle extrêmement simplifié, où le code du programme ne comporte que des affectations de variables `x = ...` ou des appels de fonctions `f(a,b,...)`, comme le code Python suivant :

```
0 d = 1
1 b = 2 * d
2 a = 3
3 d = a * b
4 print(d)
5 c = 2 * a - b
6 print(a)
7 d = c + b
8 b = 2 * a
9 print(c,d)
```

Comme un programme ne peut utiliser qu'un nombre fini de variables, on suppose dans toute la suite qu'on dispose d'une numérotation des variables, ce qui permet de représenter une variable par un entier naturel. Dans l'exemple ci-dessus, on supposera que `a` est numérotée 0, `b` est numérotée 1, etc.

Une affectation de variable est représentée par un couple `(i, [j,k,...])` constitué du numéro de la variable qui figure à gauche du symbole = et de la liste des numéros des variables qui figurent à droite. Par exemple, l'affectation `c = 2 * a - b` est représentée par le couple `(2, [0,1])`.

Un appel de fonction est représenté par un couple `(None, [j,k,...])` dont le deuxième élément est la liste des numéros des variables utilisées dans l'appel de fonction. Par exemple, l'appel `print(c,d)` est représenté par le couple `(None, [2,3])`.

Les constantes apparaissant dans les différentes instructions ne sont pas prises en compte dans cette représentation.

Le programme précédent est ainsi représenté par la liste de couples suivante :

`prog = [(3,[]), (1,[3]), (0,[]), (3,[0,1]), (None,[3]), (2,[0,1]), (None,[0]), (3,[1,2]), (1,[0]), (None,[2,3])]`

On cherche à savoir à quels moments il est utile de conserver dans les registres du processeur les valeurs des différentes variables.

Dans l'exemple du programme ci-dessus, si on s'intéresse à la variable `d`, on observe que :
- elle est initialisée en ligne 0, et cette valeur est utilisée en ligne 1 ;
- elle est modifiée en ligne 3, et la nouvelle valeur est utilisée en ligne 4 ;
- elle est modifiée en ligne 7, et la nouvelle valeur est utilisée en ligne 9.

La variable `d` a donc trois périodes de vie, qu'on peut représenter par les intervalles $]0,1]$, $]3,4]$ et $]7,9]$.

La variable `a` a une seule période de vie : l'intervalle $]2,8]$; la variable `b` n'a également qu'une période de vie : l'intervalle $]1,7]$. En effet, `b` est modifiée en ligne 8 mais sa valeur n'est pas utilisée dans la suite, donc on ne tient pas compte de l'intervalle $]8,9]$.

On dit qu'une variable est vivante pour chaque instruction d'une période de vie, et morte pour les autres.

L'algorithme suivant permet de déterminer pour chaque ligne de code si chaque variable est morte ou vivante.

- on commence par indiquer que chaque variable est morte ;
- pour chaque ligne de code, **en partant du bas** :
  - s'il s'agit d'un appel de fonction, chaque variable utilisée est marquée vivante ;
  - s'il s'agit d'une affectation, la variable affectée est marquée morte sauf si elle figure également à droite du symbole =, et toutes les variables à droite du symbole = sont marquées vivantes.

Question 17.

Recopier et compléter les lignes 0 à 4 du tableau suivant qui indique l'état de chaque variable pour chaque instruction du programme ci-dessus. (Un V indique une variable vivante, un M une variable morte.)

ligne	a	b	c	d
0				
1				
2				
3				
4				
5	V	V	M	M
6	V	V	V	M
7	V	V	V	M
8	V	M	V	V
9	M	M	V	V

Question 18. Comment interpréter le cas où en ligne 0 du tableau on trouve une variable vivante ? Le programme est-il exécutable ?

Question 19.

Pour un programme comportant n instructions et utilisant p variables numérotées de 0 à $p-1$, le tableau qui indique l'état de chaque variable pour chaque instruction peut être représenté par une liste de listes `vie` telle que pour la ligne i (avec $0 \leqslant i < n$) et la variable v (avec $0 \leqslant v < p$), `vie[i][v]` est `True` si la variable est vivante et `False` si la variable est morte.

Écrire une fonction `determineVie(prog,p)` qui prend en argument un programme et le nombre p de variables à considérer et qui renvoie le tableau `vie` correspondant.

Question 20.

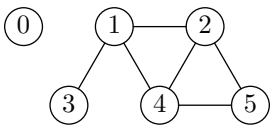
Recopier et compléter la fonction suivante `periodesVie(vie,v)` qui prend en arguments la table `vie` obtenue par la fonction `determineVie` et le numéro d'une variable et qui renvoie la liste des périodes de vie de cette variable. Ainsi, pour l'exemple du programme considéré, et la variable `d`, la fonction `periodesVie` renvoie la liste `[(7, 9), (3, 4), (0, 1)]` (l'ordre n'a pas d'importance).

```
def periodesVie(vie,v):
    n = len(vie)
    periodes = []
    i = n-1
    encours = False
    while i>=0:
        if vie[i][v]:
            if not encours:
                ...
                ...
        else:
            if ...:
                ...
                ...
        i -= 1
    return periodes
```

Partie C – graphe de coexistence

Des techniques analogues à celle présentée dans la partie B permettent de construire un graphe de coexistence des variables occupant des registres. Il s'agit d'un graphe non orienté tel que deux sommets reliés dans ce graphe correspondent à deux variables vivantes au même moment, ce qui oblige à les ranger dans des registres différents.

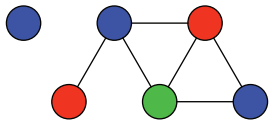
La figure ci-dessous montre un exemple de graphe de coexistence : les variables numérotées 1, 2, 4 doivent être rangées dans trois registres distincts. Mais les variables numérotées 1 et 5 peuvent être rangées dans le même registre, puisque les sommets correspondants ne sont pas reliés.



Tournez la page S.V.P.

Pour déterminer le nombre de registres nécessaire, il suffit de colorier le graphe de sorte que deux sommets adjacents soient de couleurs différentes, en utilisant le moins de couleurs possible. Ce nombre minimal de couleur est le nombre cherché de registres.

Dans l'exemple du graphe ci-dessus, trois couleurs suffisent.



Question 21.

Le problème général de la détermination du nombre minimal de couleurs nécessaire à la coloration d'un graphe est connu pour être un problème NP-complet. Que signifie l'expression « ce problème est dans la classe NP »? et que signifie l'adjonction de l'adjectif « complet »?

Dans la suite, un graphe non orienté possédant n sommets numérotés de 0 à $n-1$ est représenté par une liste `G` de n listes : `G[i]` est la liste (éventuellement vide) des sommets reliés au sommet `i`. Autrement dit, `G[i]` est la liste des voisins du sommet `i`.

L'exemple du graphe précédent est représenté par la liste `G=[ [], [2,3,4], [1,4,5], [1], [1,2,5], [2,4] ]`.

Une coloration du graphe est représentée par une liste donnant le numéro de la couleur de chaque sommet. Dans l'exemple ci-dessus, si bleu est numéro 0, rouge numéro 1 et vert numéro 2, la coloration proposée est représentée par la liste `couleur = [0,0,1,1,2,0]` .

Question 22.

Écrire une fonction `bonnesCouleurs(G,couleur)` qui prend en argument un graphe `G` et une coloration `couleur` de ses sommets et qui renvoie `True` si et seulement si la coloration est correcte, c'est-à-dire que deux sommets adjacents n'ont jamais la même couleur.

Question 23.

Il est facile d'écrire un algorithme qui permet de déterminer une coloration correcte d'un graphe, mais sans garantir qu'on utilise le nombre minimal de couleurs.

Par exemple, en partant d'un graphe dont aucun sommet n'est colorié.

1. choisir la première couleur disponible ;
2. tant qu'il existe un sommet non colorié répéter les étapes 3 à 6 ;
3. choisir un sommet s non colorié, et le colorier avec la couleur courante ;
4. répéter l'étape 5 pour tout sommet t non colorié ;
5. si t n'est adjacent à aucun sommet colorié avec la couleur courante, le colorier avec la couleur courante ;
6. choisir une nouvelle couleur ;
7. fin de l'algorithme.

Écrire une fonction `coloriage(G)` qui prend en argument un graphe et renvoie à l'aide de l'algorithme ci-dessus une coloration possible de ses sommets. On pourra utiliser (sans la recopier) la fonction auxiliaire suivante qui vérifie qu'aucun sommet de la liste `voisins` n'est de la couleur `c`.

```
def coloriable(voisins,couleur,c):
    for v in voisins:
        if couleur[v] == c: return False
    return True
```