☺ Examen d'algorithmique ☺

Pierre Coupechoux

8 novembre 2021

Les règles

Le but de cet examen est de mesurer votre niveau actuel. Autant pour moi que pour vous. Pour cette raison, je vous demande de répondre aux questions avec vos propres mots. Ça ne sert à rien d'aller chercher des réponses sur internet. Je ne vous demande pas une définition exacte, au mot près, apprise par coeur, mais plutôt une explication personnelle.

L'examen est découpé en 3 grandes parties. Dans la première, je vous pose des questions "bêtes et méchantes". Les deux autres parties sont des petits TPs : ce sont des suites de petites questions qui s'enchaînent de façon cohérente pour vous amener, petit à petit, vers un but un peu plus conséquent. La deuxième partie vise à trouver le jour de la semaine d'une date donnée, alors que la troisième partie consiste à réaliser une simulation d'un certain "jeu". Cette troisième partie est certainement un peu plus compliquée que les deux autres.

Si vous n'arrivez pas à répondre à toutes les questions de l'examen, ce n'est pas grave : il y a plus de questions que nécessaire, pour essayer de faire en sorte que personne ne s'ennuie (même si l'ensemble est réalisable en une heure ; #sansPression). **Prenez le temps de bien lire les questions** et **lisez une fois entièrement le sujet avant de commencer** ! Et j'insiste, il est important de bien tout lire ! D'ailleurs, la question numéro zéro est cachée ici : il faut rajouter un commentaire "J'ai bien lu les consignes" tout en haut du fichier, à la ligne numéro un. Merci de ne pas "spoil" vos camarades !

Je vous ai fourni un fichier "template_a_renommer_avec_son_prenom_NOM.js". Ce fichier contient quelques lignes de code incomplètes (et donc à compléter), quelques tests (que vous pouvez ici aussi compléter), ainsi que des commentaires pour organiser plus facilement vos réponses. Quand une question demande une réponse en français, il suffit de mettre cette réponse en commentaire (comme c'est fait pour la réponse à la question numéro 1). Et quand la question demande une réponse en js, il suffit d'écrire directement le code.

1 Les différents concepts

Idéalement, cette partie ne devrait pas vous prendre plus d'une heure et demi.

1.1 Les variables

Un des premiers concepts que l'on a vus est celui de **variable**.

Question 1 : **Avec vos mots**, qu'est-ce qu'une variable ? Quand va-t-on utiliser une variable ?

Question 2 : Qu'est-ce que la déclaration d'une variable ? Quel mot-clé permet de réaliser cette action ?

Question 3 : À quoi sert le symbole = ?

Question 4 : Traduire l'algorithme suivant :

- Créer une variable a qui contient le nombre 5.
- Créer une variable b qui contient le double de a.
- Afficher dans la console le résultat de la somme de a et b.

Question 5 : Comment échanger les contenus de deux variables ?

1.2 Les boucles

Un autre concept, découvert rapidement sur algoblocs, est le concept de **boucle**.

Question 6 : Qu'est-ce qu'une boucle ?

Question 7 : Utiliser une boucle pour écrire 27 fois "Il ne faut pas tricher !" dans la console.

Question 8 : Sans tester le code suivant, deviner ce qu'il va afficher.

```
let a = 7;
for(let i = -3 ; i < 5 ; i+=2) {
  console.log("J'ajoute",i);
  a += i;
}
console.log("Au final, a vaut :",a);
```

Question 9 : Écrire un programme qui affiche le résultat suivant dans la console :

- 1 et 7
- 1 et 8
- 1 et 9
- 2 et 7
- 2 et 8
- 2 et 9
- 3 et 7
- 3 et 8
- 3 et 9

1.3 Les branchements conditionnels

Le concept du "if" a été découvert dans l'exercice de lecture de boucles ; il n'était pas présent dans les exercices de base d'algoblocs.

Question 10 : À quoi sert le mot-clé "if" ?

Question 11 : Proposer une traduction en js de l'algorithme suivant :

- Créer une variable c qui contient 42.
- Écrire un message dans la console :
  - si c est plus grand que 20, le message sera "C'est grand !"
  - sinon si c vaut 20, le message sera "20 tout pile !"
  - sinon, le message sera "C'est petit !"

1.4 Les fonctions

Un des premiers concepts que l'on a utilisés, même sans le savoir au début, est celui des **fonctions**.

Question 12 : Qu'est-ce qu'une fonction ?

Question 13 : Donner des exemples de fonctions que l'on a croisées dans le cours d'algo (avant que nous sachions les écrire nous-mêmes).

Considérons la fonction suivante (qu'il ne faut pas copier dans le code) :

```
function aquali(a,b,c) {
  let d = a+2*b;
  return d + 3*c;
}
```

Question 14 : Ici, est-ce qu'on définit la fonction `aquali` ou est-ce qu'on l'utilise ?

Question 15 : Que vaut `aquali(1,2,3)` ?

Question 16 : Que vaut `aquali(aquali(1,1,1),aquali(2,2,2),aquali(3,3,3))` ? Il est fortement conseillé de détailler les calculs !

Question 17 : Écrire une fonction appelée `f1` qui prend un nombre en entrée et qui retourne l'opposé de ce nombre. Par exemple, l'opposé de 5 est -5. L'opposé de -7 est 7.

Question 18 : Écrire une fonction `f2` qui prend deux nombres en entrée et qui retourne (grâce à la fonction `f1`) l'opposé de la somme des opposés des nombres. ⚠ Interdiction de prendre des raccourcis ! Vous avez le droit de relire la consigne plusieurs fois, alors profitez-en, prenez le temps de bien lire la phrase !

1.5 Les tableaux

Le dernier concept que nous avons découvert est celui des **tableaux**.

Question 19 : Qu'est-ce qu'un tableau ?

Question 20 : Créer une variable `t` qui contient un tableau de 10 cases (valeurs au choix).

Question 21 : Quel est le numéro de la dernière case de `t` ?

Question 22 : Quel est le numéro de la troisième case de `t` ?

Question 23 : Comment retirer la dernière case du tableau `t` ?

Question 24 : Comment afficher dans la console le nombre de cases de `t` ?

Question 25 : Écrire un code qui permet d'afficher tous les nombres contenus dans le tableau, un par un.

2 TP1 - Les jours de la semaine

Le but de cette partie est de déterminer par nous-mêmes le jour de la semaine d'une date donnée. Le résultat ne sera pas parfait (historiquement, le calendrier avant 1582 fonctionnait différemment), mais il nous permettra de manipuler les concepts de la partie précédente.

Les jours de la semaine

Question 26 : Créer un tableau de 7 cases, qui contient les mots "lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi" et "dimanche". Quand nous ferons des calculs, nous allons utiliser des nombres (c'est beaucoup plus simple pour un ordinateur de manipuler des nombres ; et en vrai, il ne sait faire que ça). Les jours de la semaine seront assez naturellement numérotés de 0 à 6.

Question 27 : Comment utiliser le tableau précédent pour connaître le nom du jour de la semaine qui correspond à un nombre ? Exemple : le nom qui correspond à 3 est "jeudi".

Question 28 : Écrire une fonction `nameOfDay` qui reçoit un nombre entre 0 et 6 en entrée et qui retourne le nom du jour correspondant. On ne demande pas de vérifier que le nombre est bien compris entre 0 et 6, on suppose que c'est bien le cas. Exemple d'utilisation : `nameOfDay(3)` devra retourner `"jeudi"`.

Les années bissextiles

Les années bissextiles ont 366 jours par an, alors que les années non bissextiles n'ont que 365 jours par an. Les deux règles qui permettent de déterminer si une année est bissextile ou non sont les suivantes :

- Les années multiples de 400 sont bissextiles.
- Les années multiples de 4 **mais** non multiples de 100 sont bissextiles.

L'année 2020 était bissextile car elle est multiple de 4 mais pas de 100. L'année 2000 était aussi bissextile, car multiple de 400. En revanche, 1900 n'était pas bissextile. Déjà, 1900 n'est pas un multiple de 400. C'est bien un multiple de 4, mais aussi un multiple de 100.

Question 29 : En utilisant les règles précédentes, dire si les années de 2010 à 2021 sont bissextiles, et expliquer pourquoi. Vous pouvez présentez vos résultats sous cette forme là :

- 2009 n'est pas bissextile car elle n'est ni multiple de 400, ni multiple de 4.
- 2010 est/n'est pas bissextile parce que...

Question 30 : Écrire une fonction `isLeapYear` qui prend en entrée une année (un nombre) et qui retourne `true` si cette année est bissextile, `false` sinon. Exemples d'utilisation :

- `isLeapYear(2021)` devra retourner `false`.
- `isLeapYear(2020)` devra retourner `true`.
- `isLeapYear(2000)` devra retourner `true`.
- `isLeapYear(1900)` devra retourner `false`.

Question 31 : Écrire une fonction `nbOfDaysBetweenYears` qui prend deux années en entrée (la première sera supposée inférieure ou égale à la seconde), `year1` et `year2`. Cette fonction retournera le nombre de jours passés entre le 1er janvier de l'année `year1` matin, au 31 décembre de l'année `year2` au soir. Quelques exemples d'utilisation :

- `nbOfDaysBetweenYears(2021,2021)` retourne 365.
- `nbOfDaysBetweenYears(2020,2021)` retourne 731.
- `nbOfDaysBetweenYears(2000,2021)` retourne 8036.

Nombre de jours depuis le début de l'année

Question 32 : Écrire une fonction `nbOfDaysInYear` qui prend une année en entrée et qui retourne un tableau de 12 cases contenant le nombre de jours de chaque mois. ⚠ Attention à février ! Quand l'année est bissextile, février comporte 29 jours ; 28 sinon.

Un exemple d'utilisation : `nbOfDaysInYear(2021)` retourne `[31,28,31,30,31,30,31,31,30,31,30,31]`.

Question 33 : Écrire une fonction `numOfMonth` qui convertit un nom de mois en français ("janvier","avril", "octobre",...) en son numéro. Janvier sera le mois numéro 0, et décembre le mois numéro 11. Pour ce faire, vous pouvez commencer par créer un tableau qui contient les 12 noms des mois, et parcourir ce tableau.

Exemple d'utilisation : `numOfMonth("novembre")` retourne 10.

Question 34 : Combien de mois entiers se sont écoulés depuis le 1er janvier jusqu'au 8 novembre 2021 ?

Question 35 : En utilisant (entre autre) une boucle, compter combien de jours se sont écoulés du 1er janvier jusqu'au 8 novembre 2021 (inclus) (vous aurez aussi besoin de la fonction `nbOfDaysInYear`).

Question 36 : Écrire une fonction `nbOfDaysSinceNewYear` qui reçoit trois informations : un nombre, un mot et un autre nombre qui correspondent à une date. Par exemple, 8, "*novembre*" et 2021. Cette fonction devra retourner combien de jours se sont écoulés depuis le début de l'année, jusqu'à cette date (incluse).

Exemple d'utilisation : `nbOfDaysSinceNewYear(8,"novembre",2021)` devra retourner 312.

Le jour de la date de naissance

Nous allons nous servir d'une date arbitraire pour trouver le jour de n'importe quelle date. Cette date de référence sera le 23 juin 1912. C'était un jour numéro 6 (autrement dit, un dimanche).

Question 37 : À l'aide des fonctions écrites précédemment, calculer combien de jours se sont écoulés :

- Du 1er janvier 1912 au 23 juin 1912.
- Du 1er janvier 1912 au 31 décembre 2020.
- Du 1er janvier 2021 au 8 novembre 2021.

Question 38 : En déduire combien de jours se sont passés du 23 juin 1912 au 8 novembre 2021.

Question 39 : Sachant que le 23 juin 1912 était un dimanche, vérifier que le 8 novembre 2021 est un lundi (on ne sait jamais...).

Question 40 : Écrire une fonction `dayOfTheWeek` qui reçoit trois informations en entrée qui représentent une date (comme pour `nbOfDaysSinceNewYear`). Cette fonction devra retourner le nom du jour de la semaine correspondant à cette date, en utilisant le raisonnement des trois questions précédentes.

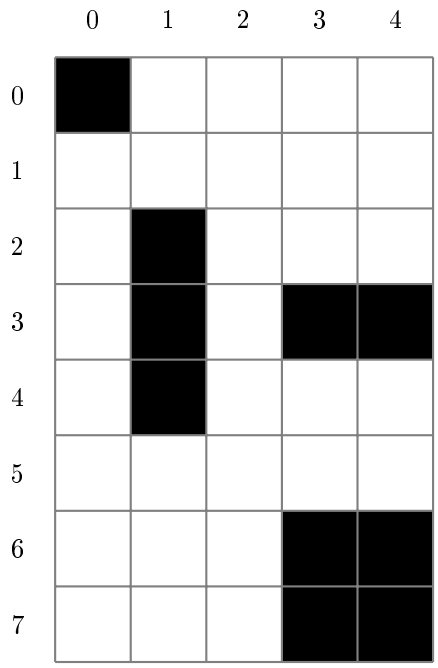
Exemple d'utilisation : `dayOfTheWeek(8,"novembre",2021)` devra retourner `"lundi"`.

3 TP2 - Le jeu de la vie

Le jeu de la vie est ce qu'on appelle un "automate cellulaire". Ici, on ne cherchera pas à détailler ce que sont ces derniers, mais nous allons essayer d'en coder un en particulier.

Dans le jeu de la vie, il y a une grille d'une certaine taille. Les cases de cette grille sont appelées des *cellules*. Chaque cellule peut être soit morte, soit vivante.

Les lignes et les colonnes de la grille seront numérotées à partir de 0, de haut en bas et de gauche à droite.



Dans ce dessin, les cellules vivantes sont dessinées en noir, et les cellules mortes sont dessinées en blanc. La cellule tout en haut à gauche (en $(0, 0)$) est vivante, alors que la cellule en haut à droite (en $(4, 0)$) est morte.

Chacune des cellules de la grille possède jusqu'à 8 cellules voisines (en haut à gauche/droite, en haut, à gauche, à droite, en bas et en bas à gauche/droite).

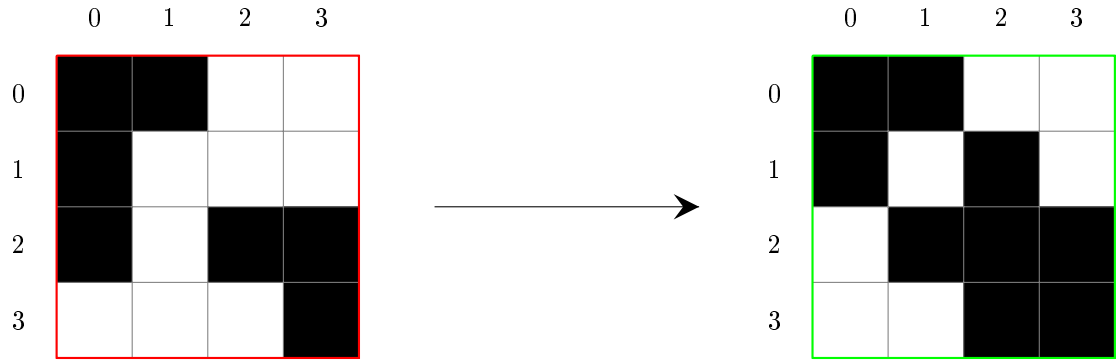
Le principe du jeu de la vie, c'est de faire évoluer les cellules. Certaines cellules vont mourir alors que d'autres vont naître (devenir vivantes). Les règles suivantes permettent de dicter l'évolution des cellules :

- Une cellule vivante reste vivante seulement si elle a 2 ou 3 voisines vivantes (ni plus, ni moins).
- Une cellule morte devient vivante si elle a exactement 3 voisines vivantes.

Une itération du jeu de la vie consiste à faire évoluer en même temps toutes les cellules de la grille. L'image suivante montre une telle itération. La grille de gauche, en rouge, représente la grille *avant* l'évolution, et celle de droite, en vert, est la même *après* évolution.

Quelques explications, pour certaines cellules :

- La cellule en $(0, 0)$ était vivante au début, et reste vivante à la fin, car elle a 2 cellules voisines vivantes (en $(0, 1)$ et en $(1, 0)$).
- Le cellule en $(0, 2)$ était vivante, mais elle meure car elle n'a qu'une seule voisine vivante.
- La cellule en $(2, 3)$ était morte, mais devient vivante car elle a trois voisines vivantes (les trois cellules en bas à droite).



Pour pouvoir répondre aux questions suivantes plus facilement, nous allons représenter les grilles comme ci-dessous. Les 1 représentent les cellules vivantes, et les 0, les cellules mortes.

1	1	0	0
1	0	0	0
1	0	1	1
0	0	0	1

1	1	0	0
1	0	1	0
0	1	1	1
0	0	1	1

Question 41 : Que donnera l'itération suivante (de la grille verte) ?

Question 42 : Comment pourrait-on, dans un premier temps, stocker **une seule ligne** d'une telle grille ?

Question 43 : Créer 4 variables qui vont stocker les 4 lignes de la grille rouge.

Question 44 : Stocker ces 4 variables dans une seule nouvelle variable.

Question 45 : Quel est le numéro de ligne de la cellule en position $(3, 2)$? Dans la grille rouge, cette cellule est vivante.

Question 46 : Comment récupérer, à partir de la variable qui contient la grille rouge entière, l'état de la cellule $(3, 2)$ (attention, il est facile de se tromper !) ?

Question 47 : Écrire une fonction `lineToString` qui reçoit une ligne d'une grille, et qui retourne une chaîne de caractères qui représente la ligne. Il est possible pour ça de s'inspirer des opérations suivantes :

```
let s = ""; // Un mot vide
s += 1; // s vaut "1"
s += 0; // s vaut "10"
s += 1; // s vaut "101"
```

Question 48 : Écrire une fonction `displayGrid` qui prend une grille en entrée et qui affiche dans la console la grille entière, grâce à la fonction de la question précédente.

Question 49 : Écrire une fonction `createEmptyArray` qui prend un nombre en entrée, et qui retourne un tableau de ce nombre de cases (avec des 0 dedans).

Question 50 : Écrire une fonction `createEmptyGrid` qui prend deux nombres en entrée, et qui retourne une grille de cette taille là (pleine de 0).

Question 51 : Écrire une fonction `evolution` qui prend en entrée une grille et qui retourne une nouvelle grille correspondant à l'évolution de celle donnée à la fonction. Par exemple, en donnant la grille rouge à la fonction, on doit obtenir la grille verte en retour.

Question 52 : Écrire une fonction `timeJump` qui prend en entrée une grille et un nombre, et qui retourne la grille obtenue après ce nombre d'évolutions.

Affichage en p5

Pour avoir un résultat plus interactif et plus ludique, on peut faire un nouveau projet en p5 pour illustrer les résultats des fonctions précédentes.

Question 53 : Écrire une fonction `drawGrid` qui reçoit une grille en entrée et qui la dessine dans un canevas, à l'aide de p5.

Question 54 : Faire en sorte que lorsque l'on appuie sur une touche, la grille évolue une fois.

Le TP se termine ici, mais libre à vous de continuer le programme. Quelques idées parmi d'autres : on peut changer l'état d'une cellule en cliquant dessus, on peut laisser le choix de la taille à l'utilisateur, on peut compter combien d'évolutions il faut pour que toutes les cellules meurent (si jamais ça arrive), on peut recoller les bords de la grille : la colonne de gauche sera voisine de la colonne de droite, on peut zommer/dézommer avec la molette de la souris, etc.