#07 : Codin'game

1) Introduction

Nous avons appris les bases du langage Javascript. Vous avez donc constaté que sa syntaxe n'est pas compliquée et qu'il est facile à mettre en œuvre. Ceci dit, la difficulté de la programmation n'est pas d'utiliser un langage mais bien de concevoir ses propres programmes en organisant les instructions et les données intelligemment pour aboutir efficacement au résultat souhaité.

Nous allons donc nous entraîner à programmer en résolvant des « jeux de programmation » sur le site Codin'game.

2) Premier programme : Onboarding

❑ Commencez par créer un compte sur le site de Codin'game (www.codingame.com).

❑ Une fois la procédure terminée, cliquez sur la petite flèche à droite de votre avatar afin de sélectionner **Settings**. Sélectionnez tout en bas des réglages la langue française pour basculer le site en français.

❑ Ensuite choisissez de résoudre votre premier jeu en cliquant sur **Entraînement** dans la barre de gauche du site, il se nomme **Onboarding**. Cliquez sur **Résoudre** pour commencer le jeu. Vous entrez alors dans le tutoriel.

❑ Lisez les consignes et choisissez le langage **Javascript** puis suivez les consignes afin d'écrire votre premier programme.

❑ Vous avez alors à gauche l'énoncé de la mission et à droite votre programme. Ce qu'il faut bien comprendre dès le début est expliqué dans le tutoriel : « Les jeux se déroulent en **tour par tour : à chaque tour, votre programme reçoit de nouvelles entrées et doit répondre par une action** ».

❑ A la fin de la procédure vous obtenez le programme suivant :

```
// game loop : une boucle infinie qui sera relancée à chaque tour de jeu

while (true)
{
    var enemy1 = readline(); // name of enemy 1 : Lecture du nom de l'ennemi n°1
    var dist1 = parseInt(readline()); // distance to enemy 1 : Lecture de sa distance

    var enemy2 = readline(); // name of enemy 2: Lecture du nom de l'ennemi n°2
    var dist2 = parseInt(readline()); // distance to enemy 2 : Lecture de sa distance

      // Enter the code here
      /* Les entrées de ce tour sont lues, on les analyse et on répond en écrivant le nom
      de l'ennemi sur lequel on doit tirer. */

      if (dist1 < dist2)
      {
            print(enemy1); // On écrit le nom de l'ennemi 1 si il est le plus proche
      }
      else
      {
            print(enemy2);  // On écrit le nom de l'ennemi 2 si il est le plus proche
      }
}
```

❑ Expliquez à quoi sert la fonction parseInt() utilisée ici ?

_ _

❑ A quoi sert le test « if (dist1 < dist2) » ?

_ _

❑ Finissez le tutoriel et soumettez votre code.

3) Power of Thor - Episode 1

❑Lancez la résolution du jeu : Power of Thor – Episode 1. Le squelette du programme est le suivant :

```
var inputs = readline().split(' ');
var lightX = parseInt(inputs[0]); // the X position of the light of power
var lightY = parseInt(inputs[1]); // the Y position of the light of power
var initialTX = parseInt(inputs[2]); // Thor's starting X position
var initialTY = parseInt(inputs[3]); // Thor's starting Y position

// game loop
while (true)
{
    var remainingTurns = parseInt(readline()); // Lecture du nombre de tours restants

    // To debug: printErr('Debug messages...');

    // A single line providing the move to be made: N NE E SE S SW W or NW
    print('SE');
}
```

Vous avez donc deux variables, **lightX** et **lightY** qui contiennent les coordonnées de l'éclair à atteindre. De même vous disposez des coordonnées initiales de Thor dans les variables **initialTX** et **initialTY**.

Il faut donc à chaque tour de jeu écrire une direction pour que Thor se déplace d'une case, jusqu'à atteindre l'éclair. Dans ce programme initial la direction est toujours **"SE"** donc cela ne peut pas fonctionner.

❑Remplacez la ligne : // To debug: printErr('Debug messages...'); par :

```
printErr("Test des variables : ", lightX, lightY);
```

❑Relancez le 1er test et regardez la sortie de la console. Vous avez là un moyen de connaître l'état de vos variables en ajoutant cette fonction où vous voulez dans votre programme.

❑Pour élaborer le programme répondez à : « A quelle condition Thor doit choisir la direction « N » ?

_ _

❑Puis répondez à cette question : « A quelle condition Thor doit choisir la direction « S » ?

_ _

❑Puis répondez à cette question : « A quelle condition Thor doit choisir la direction « E » ?

_ _

❑Puis répondez à cette question : « A quelle condition Thor doit choisir la direction « W » ?

_ _

❑Il faut donc avoir deux variables (**thorX** et **thorY**) pour lire et enregistrer la position de Thor à chaque tour. Déclarez ces deux variables avant la « game loop » et donnez leur comme valeurs initiales les coordonnées initiales de Thor.

❑Traduisez ces premières conditions dans votre programme et essayez / modifiez le. Souvenez-vous que **votre programme ne doit envoyer qu'un seul ordre de déplacement print() par tour !**

❑Une fois que votre programme passe tous les tests, essayez de l'optimiser en économisant de la place en mémoire en évitant de créer les variables thorX et thorY.

❑Vous pouvez (si vous êtes très à l'aise) rechercher une solution avec seulement 4 tests…

4) La descente

❑Lancez la résolution du jeu : La descente. Modifiez le squelette du programme comme ceci :

```
var mountainH = 0;      // hauteur de la montagne
var altMax = 0;         // altitude maximale
var mountainIndex = 0;  // index de la montagne à detruire

// game loop
while (true)
{
   for (var i = 0; i < 8; i++)
   {
      mountainH = parseInt(readline()); //Lecture de la hauteur de la montagne d'index i
      // complétez ici
   }

   print(mountainIndex); // The index of the mountain to fire on.
}
```

❑Vous devez compléter la fin de la boucle for pour trouver l'index de la plus haute montagne et l'écrire dans la variable **mountainIndex**.

❑A chaque tour de la boucle for on traite une montagne différente (index n°0 puis 1 puis 2 etc.). Quel test faire pour savoir si la montagne actuelle est la plus haute ?

_ _

❑Que doit-on faire si la montagne actuelle est pour l'instant la plus haute ?

_ _

_ _

❑La valeur de l'altitude maximale est-elle la même à chaque tour de jeu ?

_ _

❑Essayez et modifiez votre programme pour réussir tous les tests puis soumettez votre code.

5) Mars Lander - Episode 1

❑Lancez la résolution du jeu : Mars Lander – Episode 1. Dans ce jeu, beaucoup de variables sont inutiles car elles servent pour les niveaux suivants. Seules les variables vSpeed et power vous seront utiles.

Aide : pour connaître la valeur absolue d'un nombre (la distance à zéro) vous pouvez utiliser : Math.abs(la_variable). Pour gagner du temps, il existe aussi une fonction qui renvoie la plus grande valeur entre deux nombres : Math.max(nombre1, nombre2).

Aide : pour fabriquer la chaîne de caractères finale vous pouvez utiliser à la fin du tour :

```
print("0" + " " + power);
```

❑A vous de trouver les conditions à appliquer pour réaliser le calcul de la puissance (variable power) des moteurs à appliquer à chaque tour de jeu.

❑Essayez et modifiez votre programme pour réussir tous les tests puis soumettez votre code.

❑Vous pouvez (si vous êtes très à l'aise) rechercher une solution avec seulement 2 lignes à ajouter…

6) Températures

❑Résolvez le jeu en commençant par modifier le squelette du programme comme ceci :

```
var n = parseInt(readline()); // the number of temperatures to analyse
var temps = readline(); // the n temperatures expressed as integers from -273 to 5526

var result;
var tablo = temps.split(" ").map(Number); // tableau des temperatures sous forme de nombre

for (             )     // une boucle qui parcourt toutes les valeurs de tablo
{

}

print(result);
```

Aide : pour connaître la valeur absolue d'un nombre (la distance à zéro) vous pouvez utiliser : Math.abs(la_variable).

❑Construisez, essayez et modifiez votre programme pour réussir tous les tests puis soumettez votre code.

7) Des autres faciles…

Pour continuer je vous conseille d'enchaîner avec : **Chevaux de courses** et **Défibrillateurs** qui sont faciles. Mais bien évidemment, vous pouvez tous les essayer car la programmation c'est comme tout le reste : plus on en fait, meilleur on est :-) !