Sujet K4.A :

Écrivez une fonction qui prend un entier non négatif et insère des tirets ('-') entre chaque paire de chiffres impairs consécutifs, ainsi que des astérisques ('*') entre chaque paire de chiffres pairs non nuls consécutifs. Le chiffre zéro ('0') ne déclenche pas l'insertion de tirets ou d'astérisques.

Assertions :

expect(insertHyphensAndAsterisks(454793)).to.equal("4547-9-3");

expect(insertHyphensAndAsterisks(1012356895)).to.equal("10123-56*89-5");

Sujet K4.B

Écrivez une fonction qui convertit une chaîne donnée en "Jaden Case", où chaque mot commence par une majuscule, comme Jaden Smith le fait dans ses tweets.

Assertions :

expect(toJadenCase("How can mirrors be real if our eyes aren't real")).to.equal("How Can Mirrors Be Real If Our Eyes Aren't Real");

expect(toJadenCase("le soleil brille")).to.equal("Le Soleil Brille");

expect(toJadenCase("une phrase simple")).to.equal("Une Phrase Simple");

expect(toJadenCase("123 les chiffres ne changent pas")).to.equal("123 Les Chiffres Ne Changent Pas");

Sujet K4.C

Le but de cet exercice est de créer une fonction qui prend en paramètre un string *str*, et qui retourne un tableau de strings composés des mots de *str*, triés par la dernière lettre, dans l'ordre croissant. Lire le test pour plus de détails.

```
describe("Sort by Last Char", () => {
  it("should sort words (ascending) based on their last character", () => {
    expect(last('man i need a taxi up to ubud'))
        .to.deep.equal(['a', 'need', 'ubud', 'i', 'taxi', 'man', 'to', 'up']);
    expect(last('what time are we climbing up the volcano'))
        .to.deep.equal(['time', 'are', 'we', 'the', 'climbing', 'volcano', 'up', 'what']);
    expect(last('take me to semynak')).to.deep.equal(['take', 'me', 'semynak', 'to']);
  });

  it("should preserve original order when words have the same last character", () => {
    expect(last('apple banana peach grape')).to.deep.equal(['banana', 'apple', 'peach', 'grape']);
    expect(last('cat bat hat mat')).to.deep.equal(['cat', 'bat', 'hat', 'mat']);
  });
});
```


Sujet K4.C

Le but de cet exercice est de créer une fonction qui prend en paramètre un string *str*, et qui retourne un tableau de strings composés des mots de *str*, triés par la dernière lettre, dans l'ordre croissant. Lire le test pour plus de détails.

```
describe("Sort by Last Char", () => {
  it("should sort words (ascending) based on their last character", () => {
    expect(last('man i need a taxi up to ubud'))
        .to.deep.equal(['a', 'need', 'ubud', 'i', 'taxi', 'man', 'to', 'up']);
    expect(last('what time are we climbing up the volcano'))
        .to.deep.equal(['time', 'are', 'we', 'the', 'climbing', 'volcano', 'up', 'what']);
    expect(last('take me to semynak')).to.deep.equal(['take', 'me', 'semynak', 'to']);
  });

  it("should preserve original order when words have the same last character", () => {
    expect(last('apple banana peach grape')).to.deep.equal(['banana', 'apple', 'peach', 'grape']);
    expect(last('cat bat hat mat')).to.deep.equal(['cat', 'bat', 'hat', 'mat']);
  });
});
```

Sujet K4.D

De nombreux musées vous permettent de devenir membre, et pour un certain montant annuel, vous avez un accès illimité au musée.

Dans cette tâche, vous devez compléter une fonction qui détermine après combien de visites il sera plus avantageux de prendre un abonnement annuel plutôt que de payer les billets individuellement.

La fonction prend en entrée deux arguments : annual_price (prix de l'abonnement annuel) et individual_price (prix d'un billet individuel).

Tests :

```
describe("How many visits before annual pass is worth it", () => {
  it("should return the number of visits where the annual pass becomes a better option", () => {
    expect(howManyTimes(40, 15)).to.equal(3);
    expect(howManyTimes(30, 10)).to.equal(3);
    expect(howManyTimes(80, 15)).to.equal(6);
  });
});
```

Exemple de la première assertion : si l'abonnement annuel coûte 40€ et que chaque billet individuel coûte 15€, après 3 visites, il sera plus avantageux de prendre l'abonnement.

Sujet K4.E

Écrivez une fonction qui calcule la différence de chaque valeur d'une liste avec la moyenne de la liste. Les différences doivent être arrondies à deux décimales avec .toFixed(2).

```
expect(differencesFromAverage([55, 95, 62, 36, 48])).to.deep.equal([4.2, -35.8, -2.8, 23.2, 11.2]);
```

Sujet K4.F

Supprimez les doublons d'une liste d'entiers, en conservant la dernière occurrence de chaque élément (la plus à droite). Par exemple, pour l'entrée : [3, 4, 4, 3, 6, 3], il faut supprimer le 3 à l'index 0, supprimer le 4 à l'index 1 et supprimer le 3 à l'index 3.

La sortie attendue serait : [4, 6, 3]

Plus d'exemples peuvent être trouvés dans les cas de test.

```
describe("removeDuplicates", () => {
  it("should remove duplicates and keep the last occurrence", () => {
    expect(removeDuplicates([3, 4, 4, 3, 6, 3])).to.deep.equal([4, 6, 3]);
    expect(removeDuplicates([1, 2, 3, 1, 2, 3])).to.deep.equal([1, 2, 3]);
    expect(removeDuplicates([5, 1, 2, 2, 1, 5, 6, 1])).to.deep.equal([2, 5, 6, 1]);
    expect(removeDuplicates([9, 9, 9, 9])).to.deep.equal([9]);
    expect(removeDuplicates([7, 8, 9, 7, 7])).to.deep.equal([8, 9, 7]);
  });
});
```

Sujet K4.G

On vous donne un tableau d'entiers de longueur impaire, dans lequel tous les entiers sont identiques, à l'exception d'un seul nombre. Complétez la fonction qui accepte un tel tableau d’entiers, et qui renvoie ce seul nombre différent. Le tableau d'entrée sera toujours valide et comportera toujours minimum 3 éléments.

```
describe("Find the single different number in an array", () => {

  it("should return the only number that differs in the array", () => {

    expect(findUnique([1, 1, 2])).to.equal(2);

    expect(findUnique([17, 17, 3, 17, 17, 17, 17])).to.equal(3);

    expect(findUnique([5, 5, 5, 5, 2, 5, 5])).to.equal(2);

    expect(findUnique([10, 10, 10, 5, 10])) .to.equal(5);

  });

});
```

Sujet K4.H

Mon PC a été infecté par un étrange virus. Il n'infecte que mes fichiers texte et remplace des lettres aléatoires par des astérisques, par exemple : li*e th*s. Heureusement, j'ai découvert que le virus cache mes lettres censurées dans le répertoire racine.

Il serait très fastidieux de récupérer tous ces fichiers manuellement, donc votre objectif est de mettre en œuvre une fonction uncensor qui fait tout le travail automatiquement.

```
describe("uncensor", () => {
  it("should uncensor the letters and return the correct result", () => {
    expect(uncensor("*h*s *s v*ry *tr*ng*", "Tiiesae")).to.equal("This is very strange");
    expect(uncensor("A**Z*N*", "MAIG")).to.equal("AMAZING");
    expect(uncensor("xyz", "")).to.equal("xyz");
  });
});
```

Sujet K4.I

Vous avez deux tableaux : le premier contient la clé des bonnes réponses à un examen, et le second les réponses d'un étudiant. Les deux tableaux ont la même longueur. Retournez le score de l'étudiant en attribuant : +4 pour chaque bonne réponse, -1 pour chaque mauvaise réponse, +0 pour chaque réponse vide (chaîne vide).

Si le score est inférieur à 0, retournez 0.

```
describe("Score Calculation for Exam Answers", () => {

  it("should return the correct score for the given answers", () => {

    expect(calculateScore(["a", "a", "b", "b"], ["a", "c", "b", "d"])).to.equal(6);

    expect(calculateScore(["a", "a", "c", "b"], ["a", "a", "b", ""])).to.equal(7);

    expect(calculateScore(["a", "a", "b", "c"], ["a", "a", "b", "c"])).to.equal(16);

    expect(calculateScore(["b", "c", "b", "a"], ["", "a", "a", "c"])).to.equal(0);

  });

});
```

Sujet K4.J

On vous donnera un tableau contenant à la fois des entiers et des caractères.

Retournez un tableau de longueur 2 :

- a[0] représente la moyenne des dix entiers sous forme de nombre à virgule flottante.
- a[1] est une chaîne de caractères formée par la concaténation des caractères présents dans le tableau, dans le même ordre qu'ils apparaissent.

Exemple : Pour le tableau ['u', ' ', 'd', '1', 'i', 'w', '6', 's', 't', '4', 'a', '6', 'g', '1', '2', 'w', '8', 'o', '2', '0'], le résultat attendu serait [3.6, "udiwstagwo"]

Sujet K4.K

Calculez la capacité minimale du tramway pour que le nombre de passagers à bord ne dépasse jamais cette capacité. À chaque arrêt, tous les passagers sortent avant l'entrée des nouveaux passagers.

Pour votre fonction donc, le premier paramètre le nombre de personnes qui descendent à chaque arrêt, et le deuxième est le nombre qui montent à cet arrêt. Ces deux tableaux auront toujours la même longueur. Voir les tests pour plus de détails.

```
describe("Tram Minimum Capacity", () => {

  it("should return the required minimum capacity", () => {

    expect(minTramCapacity([0, 2, 4, 4], [3, 5, 2, 0])).to.equal(6);

  });

});
```

Explication :

Le nombre de passagers dans le tram avant d'arriver est de 0.

- À la première station, 3 passagers montent dans le tram, et le nombre de passagers dans le tram devient 3.
- À la deuxième station, 2 passagers descendent du tram (1 passager reste à l'intérieur). Puis 5 passagers montent dans le tram. Il y a maintenant 6 passagers dans le tram.
- À la troisième station, 4 passagers descendent du tram (2 passagers restent à l'intérieur). Puis 2 passagers montent dans le tram. Il y a maintenant 4 passagers dans le tram.
- Enfin, tous les passagers restants dans le tram descendent lors de la dernière station. Il n'y a plus de passagers dans le tram, ce qui est conforme aux contraintes.