Java & Algorithme – Test final

Nom & Prénom : _____

Vous avez **90 minutes** pour répondre à toutes les questions. Prenez votre temps, lisez attentivement l'énoncé et tentez de bien le comprendre avant d'écrire des algorithmes.
Lisez bien tous les exercices avant de commencer : vous pouvez traiter les exercices dans l'ordre que vous souhaitez.
A chaque question, le nombre de points, ainsi que le temps recommandé à y consacrer est indiqué.
Les algorithmes doivent être écrits en Java.

1 PETITE MISE EN ROUTE (/6) – 20 MINUTES

1.1 Que font ces algorithmes ?

1.

```
public static int AlgoMystere1(int[] r)
{
   int h = r[r.length-1];

   for(int c=0; c<r.length-1; c=c+1)
   {
      if(r[c] < h)
      {
         h = r[c];
      }
   }

   return h;
}
```

2.

```
public static int AlgoMystere2(int[] r, int a)
{
   int h = 0;

   for(int c=0; c<r.length; c=c+1)
   {
      if(r[c] > a)
      {
         h = h+1;
      }
   }

   return h;
}
```

3.

```
public static int AlgoMystere3(int r, int a)
{
   for(int c=0; c<=r; c=c+1)
   {
      int v=1;
      for(int h=1; h<=a; h=h+1)
      {
         v=v*c;
      }

      if(v>r)
      {
         return (v-1);
      }
   }

   return -1;
}
```

4.

```
public static int[] AlgoMystere4(int[] a, int[] b)
{
   if(a.length == b.length)
   {
      int[] c = new int[a.length];

      for(int d=0; d<a.length; d=d+1)
      {
         if(d%2==0)
         {
            c[d] = a[d];
         }
         else
         {
            c[d] = b[d];
         }
      }
      return c;
   }
   return null;
}
```

1.2 Pourquoi ces algorithmes vont (ou peuvent) bugger, et comment l'éviter ?

1. Calcul, dans le tableau c, de la somme des cases deux à deux de a.

```
public static int[] Somme2a2(int[] a)
{
   int[] c = new int[a.length];

   for(int d=0; d<a.length; d=d+1)
   {
      c[d] = a[d] + a[d+1];
   }
   return c;
}
```

2. Rechercher l'élément maximal du tableau a.

```
public static int RechercheMax(int[] a)
{
   int c=0;

   int b=0;
   while(b < a.length)
   {
      if(a[b] >= c)
      {
         c=a[b];
      }
      else
      {
         b=b+1;
      }
   }
   return c;
}
```

3. Réaliser le produit de tous les elements du tableau a.

```
public static int Produit(int[] a)
{
   int c=1;

   for(int b=0; b<a.length; b=b+1)
   {
      c=c*a[b];
   }
   return c;
}
```

1.3 Comment corriger ces algorithmes pour qu'ils compilent et aient du sens ?

Je ne vous demande pas de réécrire tous les algorithmes, mais seulement de corriger les parties qui ne vont pas.

1. Fusion, dans le tableau c, des tableaux a et b.

```
public static int FusionTableaux(int[] a)
{
   int[] c = new int[a.length + b.lenght];

   for(int cpt=0; cpt<a.length; cpt=cpt+1)
   {
      c[cpt]=a[cpt];
   }

   for(int cpt=0; cpt<b.length; cpt=cpt+1)
   {
      c[cpt]=b[cpt];
   }

   return c;
}
```

2. Calcul de la moyenne des éléments du tableau a.

```
public static double Moyenne(int[] a)
{
   int c = 0;

   for(cpt=0; cpt<a.length; cpt=cpt+1)
   {
      c=a[cpt]+c;
   }

   double moyenne = c/a.length;
}
```

2 FAIRE UNE SUPER EQUIPE DE SUPER-HEROS (/7) – 35 MINUTES

On possède une équipe de 10 super-héros, chacun avec leurs points forts et leurs points faibles. La Terre est en danger, et ils doivent partir la défendre contre des créatures de l'espace assez agressives. Pour ce faire, nos super-héros possèdent un robot géant : le MégaZordExtraBattle+.

Le problème est que l'ascenseur qui permet d'accéder à la cabine de pilotage ne peut pas supporter plus de 300kg, et que l'on a juste le temps d'envoyer une seule équipe par cet ascenseur (car, je le rappelle, la Terre est vraiment en danger et il faut se dépêcher un peu). Il faut donc choisir quels super-héros vont pouvoir monter dans la cabine de pilotage pour combattre le monstre.

Chaque super-héros est numéroté de 0 à 9. On possède un tableau Poids (tableau d'entiers) qui permet de connaître, en kilo, le poids de chaque héros : la case 0 donne le poids du héros n°0, la case 1 donne le poids du héros n°1, etc...

On possède aussi un tableau Puissance (tableau d'entiers), qui donne la puissance de chaque héros : la case 0 donne la puissance du héros n°0, la case 1 donne la puissance du héros n°1, etc...

Le but sera de proposer un programme qui permet de trouver l'équipe pesant 300kg ou moins et possédant la plus grande puissance possible.

1. On va tout d'abord se limiter à 4 héros :
.Bioman force rouge, 85 kg, puissance de 5
.Batman, 71 kg, puissance de 7
.Iron Man, 110kg, puissance de 12
.Kungfu Panda, 135kg, puissance de 3.

On cherche à trouver la meilleure équipe **pesant 210 kilos ou moins (juste pour l'exemple).**
Pour résoudre ce problème, vous allez devoir tracer un tableau établissant toutes les combinaisons possibles de super-héros, comme montré ci-dessous :

Bioman force rouge	**Batman**	**IronMan**	**KungFu Panda**	Poids	Puissance
0	0	0	0	0	0
0	0	0	1	135	3
0	0	1	0	110	12
0	0	1	1	245 **X**	15
0	1	0	0	71	7

A chaque ligne du tableau correspond une composition d'équipe que l'on teste. Un 0 dans la colonne du super-héros signifie que le super-héros ne fait pas partie de l'équipe ; un 1 signifie qu'il fait partie de l'équipe. A droite, on affiche le poids total de l'équipe (une croix montre les poids dépassant la limite autorisée – donc les équipes que l'on doit ignorer) et la puissance de l'équipe.

Terminez ce tableau sur votre copie, et donnez la meilleure équipe avec ces 4 super-héros sans dépasser les 210 kg.

2. Combien d'équipes avez-vous dû tester en tout ? Trouvez une relation entre le nombre d'équipes à tester et le nombre de super-héros que l'on a dès le début (regardez du côté des puissances de 2). Combien d'équipes devrez-vous tester pour le problème original avec 10 super-héros ?

3. **Lisez l'annexe 2**, portant sur la conversion d'un nombre binaire en nombre décimal. A chaque ligne du tableau, vous pouvez faire correspondre un nombre binaire. Par exemple, à la quatrième ligne du tableau (représentant la combinaison 0011), on peut faire correspondre le nombre 3.
A chaque ligne de votre tableau, faîtes correspondre un nombre entier en convertissant la combinaison d'équipe en nombre décimal.

4. **Dans notre exemple, on peut donc facilement** tester toutes les combinaisons d'équipe en parcourant, avec un compteur, tous les nombres entre 0 (qui s'écrit 0000) et 15 (qui s'écrit 1111) et en convertissant ce nombre en nombre binaire permettant ainsi d'obtenir une combinaison d'équipe.
Par exemple, la 14° équipe testée correspond au nombre 14, qui en binaire donne 1110, ce qui veut dire que l'on teste l'équipe composée de Bioman force rouge, Batman et IronMan.

Lisez l'annexe 1, et proposez une fonction **ConversionBinaire** qui prend en paramètre un nombre entre 0 et 1023 (ce nombre devrait vous rappeler quelque chose en rapport avec la question 2) et convertit ce nombre en binaire. Votre fonction renverra le résultat sous forme d'un tableau d'entiers de 10 cases composées de 1 et 0.

Par exemple, si j'écris :

```
int[] tab = ConversionBinaire(378) ;
```

J'aurais dans tab :

0	1	0	1	1	1	1	0	1	0

Car 378 s'écrit 0101111010 en binaire.

5. Ecrivez un programme qui teste toutes les combinaisons d'équipe possibles entre les 10 super-héros, et qui affiche à la fin la composition de l'équipe la plus puissante et ne dépassant pas les 300 kg. Pour ce faire, vous parcourrez tous les nombres entiers entre 0 et x (à vous de déterminer x), et convertissez chaque nombre en un nombre binaire donnant une composition

d'équipe à tester.

Votre programme ne sera pas une fonction (écrivez directement le code sur la feuille, sans faire de fonction). **Vous avez le droit d'utiliser la fonction ConversionBinaire (même si vous n'avez pas répondu à la question précédente), et l'on considère que les tableaux Poids et Puissance sont déjà déclarés et initialisés.**

6. Imaginons maintenant que la seule chose qui importe, c'est d'envoyer le plus de super-héros possible dans le robot (avec toujours la limite de poids). Comment procéderiez-vous dans ce cas (**pas besoin d'écrire de code**, réfléchissez juste à la question et proposez une réponse concise) pour résoudre le problème ?

3 BIENVENUE A LA STAR ACADEMY (/7) – 35 MINUTES

Bienvenue à la Star Academy ! Vous êtes en charge de la mise en place du programme de vote par Internet, grande nouveauté de la StarAc' de cette année. Votre but sera de gérer des listes de votes et de les trier selon différents critères. Chaque vote est pour l'un des cinq candidats de l'émission : tapez 0 pour sauver Kevin, tapez 1 pour sauver Jimmy, tapez 2 pour sauver Kimberley-Jennifer, tapez 3 pour sauver Micheline-Loana, et tapez 4 pour sauver Marcel-Brandon.

Dans la suite, même si vous ne répondez pas à une question vous demandant d'écrire une fonction, vous pourrez tout de même utiliser cette fonction pour répondre à une autre question.

Voici la structure de données Vote, qui permet de définir une liste chaînée de Vote :

```
class Vote
{
        int adresse_votant ;
        int vote_candidat ;
        Vote suivant ;
}
```

Le champ adresse_votant est un entier qui permet de retrouver l'adresse IP de la personne votante. Le champ vote_candidat est un entier de 0 à 4 donnant le numéro du candidat pour qui la personne a voté.

1. On dispose d'une liste de votes qui, au fur et à mesure que les ~~imbéciles~~ téléspectateurs votent sur Internet, grandit. Lorsque l'on ajoute un nouveau Vote à notre liste de votes, on souhaite s'assurer que la personne qui a voté n'a pas déjà voté.

Pour ce faire, écrivez une fonction **AjouterVote** qui prend en paramètre un Vote, et une liste de

Vote (sous forme de la tête de liste). Votre fonction devra parcourir la liste, et si l'adresse du votant n'apparait pas dans la liste, ajouter le nouveau vote à la liste (où vous souhaitez).

Le prototype de la fonction sera le suivant :

```
public static void AjouterVote(Vote tete_de_liste, Vote vote_a_ajouter)
```

2. Ecrivez une fonction **TailleDeListe**, qui prend en paramètre la tête de la liste, et renvoie la taille de la liste.

3. On va tenter de trier la liste en passant par un tableau (ne paniquez pas !). Ecrivez une fonction **ListeVersTableau** qui prend en paramètre la tête de la liste, et renvoie en sortie un tableau de Vote : dans la première case du tableau, on aura le premier élément de la liste, dans la seconde case du tableau, on aura le second élément de la liste, etc… Vous aurez certainement besoin de la fonction TailleDeListe pour construire le tableau au début.

Le prototype de la fonction sera le suivant :

```
public static Vote[] ListeVersTableau(Vote tete_de_liste)
```

4. Ecrivez une fonction **TriTableau** qui prend en paramètre un tableau de Vote, et trie ce tableau selon les adresse_votant croissantes.

5. Ecrivez une fonction **TableauVersListe** qui prend en paramètre un tableau de Vote et renvoie une liste de Vote. Votre fonction devra simplement parcourir le tableau, et lier le premier élément au second élément du tableau (c'est-à-dire que le suivant du premier élément sera égal au second élément), lier le second élément au troisième, le troisième au quatrième, … et renvoyer, en sortie, la tête de la nouvelle liste ainsi créée (le premier élément).

Le prototype de la fonction sera le suivant :

```
public static Vote TableauVersListe(Vote[] tableau_vote)
```

6. Grâce à tout ça, écrivez une fonction **TriListe** qui prend en paramètre une liste de Vote et trie cette liste selon les adresse_votant croissantes (il faudra simplement appeler des fonctions déjà écrites précédemment).

Maintenant, on souhaite s'intéresser à qui a perdu et va partir de l'émission.

7. Ecrivez une fonction **APerdu** qui prend en paramètre la liste de vote et affiche à l'écran qui a perdu (c'est-à-dire qui a totalisé le moins de vote).

8. Vous faîtes en réalité partie de l'AASAEPSS (l'Association Anti Star Academy Et Pro Secret Story), et vous souhaitez saboter cette émission pour que les votes ne puissent pas êtes diffusés. Pour ce faire, vous devez modifier un tout petit peu la liste avant que celle-ci ne soit envoyée vers la fonction APerdu.
Que feriez-vous sur la liste pour faire bugger la fonction APerdu et faire en sorte qu'elle ne s'arrête jamais ?

4 ANNEXE1 : CONVERTIR UN NOMBRE DECIMAL EN BINAIRE

Ceci n'est pas un exercice du contrôle !

Pour convertir un nombre décimal (un nombre classique) en nombre binaire, il faut procéder tel quel : si le nombre est un multiple de 2, alors le chiffre des unités du nombre binaire (le chiffre complètement à droite) sera 0, sinon, ce sera 1. On divise ensuite le nombre par 2 (division entière) et on continue.

Si le nouveau nombre est un multiple de 2, alors le second chiffre du nombre binaire sera 0, sinon ce sera 1. On divise ensuite le nombre par 2 (division entière) et on continue.

Si le nouveau nombre est un multiple de 2, alors le troisième chiffre du nombre binaire sera 0, sinon ce sera 1. On divise ensuite le nombre par 2 (division entière) et on continue.

etc...

On arrête une fois que le nombre divisé par 2 donne 0.

Par exemple, si je souhaite convertir 44 en nombre binaire :

.44 est un multiple de 2, donc le premier chiffre du nombre binaire sera **0**.

Je divise 44 par 2, ce qui me donne 22.

.22 est un multiple de 2, donc le second chiffre du nombre binaire sera **0**.

Je divise 22 par 2, ce qui me donne 11.

.11 n'est pas un multiple de 2, donc le troisième chiffre du nombre binaire sera **1**.

Je divise 11 par 2 (division entière), ce qui me donne 5.

.5 n'est pas un multiple de 2, donc le quatrième chiffre du nombre binaire sera **1**.

Je divise 5 par 2, ce qui me donne 2.

.2 est un multiple de 2, donc le cinquième chiffre du nombre binaire sera **0**.

Je divise 2 par 2, ce qui me donne 1.

.1 n'est pas un multiple de 2, donc le sixième chiffre du nombre binaire (celui le plus à gauche) sera **1**.

Je divise 1 par 2, ce qui me donne 0, fin.

Le nombre **44** s'écrit donc, en binaire, **101100**.

5 ANNEXE 2 : CONVERTIR UN NOMBRE BINAIRE EN DECIMAL

Ceci n'est pas un exercice du contrôle !

Pour convertir un nombre binaire en décimal, il suffit de parcourir les 1 et les 0 du nombre binaire, et les multiplier par les puissances de 2 convenables. On multiplie le premier chiffre (celui le plus à droite) par 2 à la puissance 0 (c'est-à-dire 1), le second chiffre par 2 à la puissance 1 (donc 2), le troisième chiffre par 2 à la puissance 2 (donc 4), etc… et on somme tous les résultats.

Par exemple, le nombre 1101 0011 est égal à :
1 x 2 puissance 0 = 1 x 1 = 1 +
1 x 2 puissance 1 = 1 x 2 = 2 +
0 x 2 puissance 2 = 0 x 4 = 0 +
0 x 2 puissance 3 = 0 x 8 = 0 +
1 x 2 puissance 4 = 1 x 16 = 16 +
0 x 2 puissance 5 = 0 x 32 = 0 +
1 x 2 puissance 6 = 1 x 64 = 64 +
1 x 2 puissance 7 = 1 x 128 = 128

Ce qui donne en tout **211**. Voici la conversion décimal/binaire de tous les nombres entre 0 et 15 :

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111