POLYTECHNIQUE MONTRÉAL

# Questionnaire examen intra

# INF1005A

| *Identification de l'étudiant(e)* | | |
|---|---|---|
| **Nom :** | **Prénom :** | |
| **Signature :** | **Matricule :** | **Groupe :** |

| *Réservé* | |
|---|---|
| 1. | /2.5 |
| 2. | /4.0 |
| 3. | /3.0 |
| 4. | /4.0 |
| 5. | /6.5 |
| **TOTAL :** | **/20** |

| *Sigle et titre du cours* | | *Groupe* | *Trimestre* |
|---|---|---|---|
| **INF1005A – Programmation procédurale** | | **Tous** | **20161** |
| ***Professeurs*** | | ***Local*** | ***Téléphone*** |
| **Martine Bellaïche – responsable, et** | | **M-3414** | **4679** |
| ***Jour*** | ***Date*** | ***Durée*** | ***Heures*** |
| | | **2 h 00** | **18 h 00 – 20 h 00** |

| *Documentation* | *Calculatrice* | |
|---|---|---|
| ☒ Aucune | ☒ Aucune | **Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.** |
| ☐ Toute | ☐ Toutes | |
| ☒ Voir directives particulières | ☐ Non programmable | |

***Directives particulières***

- Ne recopiez pas les déclarations ni les instructions déjà fournies dans le questionnaire.
- Vous n'avez pas à écrire de commentaires ni d'en-têtes.
- Le code écrit doit être complet, c'est-à-dire qu'il est interdit d'utiliser les points de suspension (…) ou une phrase comme « Cette portion de code est répétée x fois ».
  - Au besoin, utilisez le verso de chaque page pour vos calculs ou si vous manquez d'espace pour vos réponses.
- Ne faites que les validations d'entrées demandées.
- Ne détachez pas les pages de cet examen, à l'exception de l'annexe.

***Important***

Cet examen contient **5** questions sur un total de **16** pages **(excluant cette page)**

La pondération de cet examen est de **35** **%**

Vous devez répondre sur : ☒ le questionnaire ☐ le cahier ☐ les deux

Vous devez remettre le questionnaire : ☒ oui ☐ non

**L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.**

## Fonctions interdites

`break(), continue(), error(), exist(), exit(), find(), max(), mean(), min(), numel(), quit(), return(), sort(), strfind(), strsplit(), struct, sum(), true(), unique().`

## Question 1 (2 points)

Cochez la bonne réponse (sans justification).

0.25 points / réponse

| | Vrai | Faux |
|---|---|---|
| 1) Si `ch1 = 'Bonjour'` et `ch2 = 'Monsieur'`, les instructions `strcat(ch1,ch2)` et `[ch1 ' ' ch2]` donnent le même résultat. | | F |
| 2) Soit la matrice `M = [11 23 4 5; 1 54 23 12; 8 4 30 0]`, l'instruction `size(M,1)+length(M)` donne 7. | V | |
| 3) Si `A ='Je'` et `B ='Tu'`, les deux instructions `strcmp(A,B)` et `isequal(A,B)` donnent le même résultat. | V | |
| 4) Soient B et C deux matrices ayant le même nombre de lignes, l'instruction `[B(1:2:end) C(2:2:end)]` permet de concaténer les colonnes impaires de B et les colonnes paires de C. | | F |
| 5) Soit la matrice `H =[1 2 1; 4 0 6; 0 9 4]`, l'instruction `H(H>=1 & H<=4)` retourne la colonne `[1 4 2 1 4]`. | V | |
| 6) L'instruction `floor(1 + rand(1,3)*9)` génère 3 entiers aléatoires entre 1 et 10. | | F |
| 7) Soient `X` et `Y` deux nombres entiers de signes différents, les fonctions `mod(X,Y)` et `rem(X,Y)` retournent le même résultat | | F |
| 8) Si `A =2*ones(3,4)` et `B =3*eye(4)-2`, l'instruction `A(1,:).* B(:,1)'` donne un résultat sans erreur. | V | |

## Question 2 (5 points)

Soient un ensemble de cellules `C` de dimension `2 x 4` et le tableau d'enregistrements `Enregi` qui contient un seul enregistrement ayant les champs `nom` et `val` suivants:

```
Enregi.nom = 'RHC';
Enregi.val = {'CH' [101 106 88]};
```

$$C = \begin{Bmatrix} 'RA' & -3 & \begin{bmatrix} 4 & 2 \\ 2 & 0 \end{bmatrix} & -3 \\ '2' & 4 & 'wad' & \begin{Bmatrix} Enregi & 'adi' \\ \begin{Bmatrix} \begin{bmatrix} 0.5 & -0.5 \\ -0.5 & -1.5 \end{bmatrix} & 'H' \end{Bmatrix} & 6 \end{Bmatrix} \end{Bmatrix}$$

Chaque sous-question est indépendante et considère l'ensemble de cellules *C* non modifié.
L'utilisation des structures de répétitions et décision est interdite.

2.1 Donnez l'affichage des instructions suivantes, si les instructions ne permettent pas l'affichage, indiquez qu'il y a une erreur :

1.
```
disp(C{3} + C{5}==C{2,4}{2,1}{1}.*2);
```

**1 1**
**1 1**

**0.5 points**

2.
```
C(:,[1 end])= {10};
disp(C)
```

**[10] [-3] [2×2 double] [10]**
**[10] [ 4] 'wad' [10]**

**0.5 points**

3.
```
z=length(C)
for i=1:z
    if i==2
       z=2;
       disp('Fin');
    end
    disp(i);
end
```

**1** **0.5 points**
**Fin**
**2**
**3**
**4**

4.
```
C(2,4)=[];
disp(C);
```

**Erreur** **0.5 points**

5.
```
celldisp(C{2,4}{2});
```

**ans{1} =** **0.5 points**

**0.5000 -0.5000**
**-0.5000 -1.5000**

**ans{2} =** **0.5 points**

**H**

6.
```
C{1}= [C{1}(1), lower(C{1}(2)), C{6}(1:2), ' ', C{2,4}{2}{2},
C{2,4}{3}(1:2) ];
disp(C{1});
```

**Rawa Had** **0.5 points**

2.2 Avec une seule affectation et en utilisant l'indexation de l'ensemble de cellules `C`, modifiez la cellule de façon à avoir des 0 aux positions paires. On obtiendrait :

$$C = \begin{Bmatrix} 'RA' & -3 & \begin{bmatrix} 4 & 2 \\ 2 & 0 \end{bmatrix} & -3 \\ 0 & 0 & 0 & 0 \end{Bmatrix}$$

**`C(2:2:end)={0}`** **0.5 points**

2.3 Modifier la cellule `C` d'origine en ajoutant un nouvel enregistrement au tableau d'enregistrements. Les informations liées au nouvel enregistrement sont définies comme suit :

```
nom = 'AWN';
val = {'AR' [12 15]};
```

```
C{8}{1}(2).nom = 'AWN';
```
**0.5 points**

```
C{8}{1}(2).val = {'AR' [12 15]};
```
**0.5 points**

## Question 3 (6 points)

Le programme ci-dessous permet de traiter le menu suivant comportant 4 choix :

```
1. Entrée de la matrice carrée A
2. Stockage par ligne des nombres pairs dans un ensemble de cellules
3. Affichage par ligne de la moyenne des nombres pairs
4. Quitter
```

Le programme réaffiche continuellement le menu tant et aussi longtemps que l'utilisateur ne choisit pas de quitter. L'utilisateur peut choisir de quitter à tout moment.

**Choix 1** : Le programme demande d'abord à l'utilisateur de rentrer une matrice carrée **A** (non un scalaire). Le programme doit afficher un message et redemander d'entrer la matrice **A**, tant et aussi longtemps que les conditions sur la matrice ne sont pas vérifiées. **Le Choix 1 doit être fait avant les choix 2 et 3**.

**Choix 2** : Par la suite, le programme détermine sur chaque ligne de la matrice **A** les nombres qui sont pairs et les stocke dans un ensemble de cellules **C**. L'ensemble de cellules **C** aura le même nombre de lignes que la matrice **A** et chaque ligne de **C** aura un seul vecteur contenant tous les nombres pairs se trouvant sur la ligne correspondante dans la matrice **A.** Si une ligne de **A** ne contient aucun nombre pair, la ligne correspondante dans l'ensemble de cellules **C** aura un vecteur vide.

Par exemple pour la matrice carrée **A** suivante, l'ensemble de cellules **C** sera :

$$A = \begin{bmatrix} 2 & 3 & 4 \\ 5 & 11 & 9 \\ 9 & 10 & 13 \end{bmatrix} \qquad C = \begin{Bmatrix} [2 \quad 4] \\ [\,] \\ [10] \end{Bmatrix}$$

**Le Choix 2 doit être fait avant le choix 3**.

**Choix 3** : A la fin, le programme parcourt chaque ligne de l'ensemble de cellules **C** et affiche avec fprintf() le numéro de ligne, le nombre d'éléments pairs, s'il y a lieu, et la moyenne de ces nombres pairs avec **2 chiffres après la virgule**.

Par exemple pour la même matrice carrée **A** précédente, le programme doit afficher les messages suivants :

```
Sur la ligne 1, il y a 2 nombres pairs donnant une moyenne de 3.00
Sur la ligne 2, il n'y a pas de nombre pair
Sur la ligne 3, il y a 1 nombres pairs donnant une moyenne de 10.00
```

**Choix 4** : Le programme doit quitter et afficher un message d'adieu.

Si l'utilisateur fait un choix invalide, le programme le lui signale et ré-affiche le menu.

Une ébauche de programme a été proposée. Malheureusement il est incomplet. On vous demande de compléter le programme **seulement aux sections incomplètes (**en lignes pointillées**),** afin qu'il respecte le menu et les conditions établies dans chaque choix.

```
clear all;
clc;

%Construction du menu
menu = sprintf('\n1. Entrée de la matrice carrée A \n2. Stockage par
                ligne des nombres pairs dans un ensemble de cellules
                \n3. Affichage par ligne de la moyenne des nombres
                pairs \n4. Quitter\n');


%Initialisations
```

```
……

……

……
while continuer

    ……

    choix = input('Faites votre choix: ');

    switch choix
        case 1
            M = input ('Veuillez entrer une matrice carrée : ')

            ……


                disp(' Matrice incorrecte! ');

                ……

            end

            ……

            %Pour réinitialiser à chaque entrée d'une nouvelle matrice
            C = {};
            option2 = 0;

        case 2

            if ……

                disp('Attention, il faut d''abord choisir l''option 1');
            else

               for ……

                    C{i,1} = []; %Initialise chaque ligne de C à vide

                    %Stockage dans la cellule C des nombres pairs de
                     chaque ligne de A
```

```
                ……

                    if rem(M(i,j),2) == 0

                        C{i,1} = ……
                    end

                ……

            end

            option2 = 1 ;
        end

    case 3
        if ……

            disp('Attention, il faut d''abord choisir les options 1
                  et 2');

        else
            for i = 1:size(C,1)

                if …….

                    fprintf('Sur la ligne %d, il n''y a pas de
                             nombre pair \n', i);
                else
                    %Pour calculer la moyenne des nombres pairs sur
                     chaque ligne de C

                    nb_pairs = ……

                    ……

                    for j = 1:length(C{i,1})

                        somme = ……
                    end

                    moyenne = ……

                    %Pour afficher la moyenne des nombres pairs sur
```

**chaque ligne de C**

```
                    fprintf(.....
                                .....
                                .....);
                end
            end
        end

    case 4

        .....

        disp('Au plaisir!')

    otherwise
        disp('Attention ce choix est invalide!')
end
end
```

**Réponse :**

```
clear all;
clc;

%Construction du menu
menu = sprintf('\n1. Entrée de la matrice carrée A \n2. Stockage par
                ligne des nombres pairs dans un ensemble de cellules
                \n3. Affichage par ligne de la moyenne des nombres
                pairs \n4. Quitter\n');

%Initialisations
continuer = 1;      0.25 points
option1 = 0;        0.25 points
option2 = 0;        0.25 points

while continuer
```

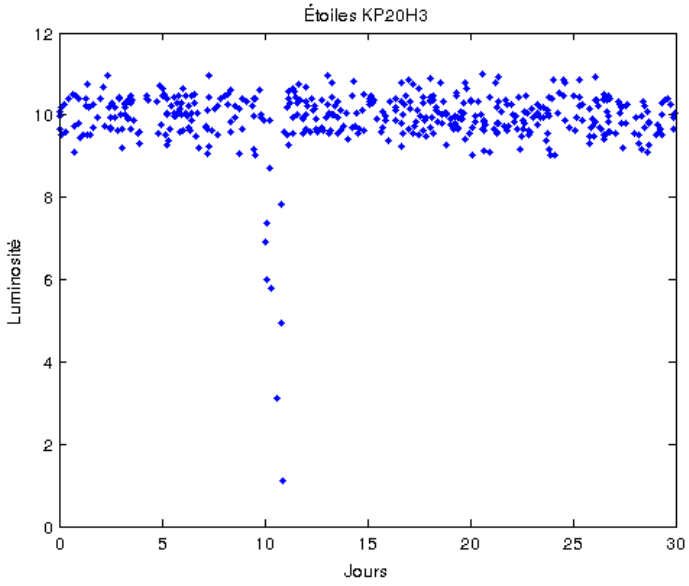
```
disp(menu);  0.25 points
choix = input('Faites votre choix: ');

switch choix
    case 1
        M = input ('Veuillez entrer une matrice carrée : ')
        while size(M,1)<=1 || size(M,2)<=1 || size(M,1)~=size(M,2)
                1 point (0.25 pour while et 0.75 les 3 conditions)

            warning (' Matrice incorrecte! ');
            M = input ('Veuillez entrer une matrice carrée : ');
                                                    0.25 points
        end
        option1 = 1;  %Pour activer l'option 1 0.25 points

        %Pour réinitialiser à chaque entrée d'une nouvelle matrice
        C = {};
        option2 = 0;

    case 2
        if option1 == 0     0.25 points
            disp('Attention, il faut d''abord choisir l''option 1');
        else
            for i = 1:size(M,1)      0.25 points
                C{i,1} = [];

                %Stockage dans la cellule C des nombres pairs de
                 chaque ligne de A
                for j = 1:size(M,2)  0.5 points (for correct et end)
                    if rem(M(i,j),2) == 0
                        C{i,1} = [C{i,1}  M(i,j)];  0.25 points
                    end
                end
            end
            option2 = 1; %Pour activer l'option 2
        end

    case 3
        if option2 == 0  0.25 points
            disp('Attention, il faut d''abord choisir les options 1
                  et 2');
```

```
            else
                for i = 1:size(C,1)
                    if isempty (C{i,1})  0.25 points
                        fprintf('Il n''y a pas de nombres pairs sur la
                                ligne %d\n',i);
                    else
                        %Pour calculer la moyenne des nombres pairs sur
                         chaque ligne de A
                        nb_pairs = length(C{i,1});   0.25 points
                        somme = 0;       0.25 points
                        for j = 1:length(C{i,1})
                            somme = somme + C{i,1}(j);   0.25 points
                        end
                        moyenne = somme/nb_pairs;  0.25 points

                        %Pour afficher la moyenne des nombres pairs sur
                         chaque ligne de C
                        fprintf('Sur la ligne %d, il y''a %d nombres
                                pairs donnant une moyenne de %.2f\n',
                                nb_pairs, i, moyenne);   0.5 points
                    end
                end
            end

        case 4 % Pour quitter le programme
            continuer = 0;  0.25 points
            disp('Au plaisir!')

        otherwise
            disp('Attention ce choix est invalide!')
    end
end
```

Question 5

## Question 4 (7 points)

Le télescope Kepler est un télescope de la NASA destiné à la détection d'exoplanètes (planètes en dehors de notre système solaire). Un projet en ligne (www.planethunters.org) permet à tout le monde de collaborer à la détection en analysant les mesures de luminosité d'étoiles susceptibles d'avoir en orbite une exoplanète.




Dans la figure ci-dessus, on voit les mesures de luminosité pour l'étoile KP20H3 sur 30 jours. Plusieurs mesures sont étonnamment faibles vers le jour 10 ce qui témoigne d'un probable passage d'une planète devant l'étoile.

On vous demande de développer des parties d'un programme existant pour gérer et manipuler les mesures réalisées sur les `N=150 000` étoiles suivies par le télescope Kepler.

Deux structures de données sont utilisées : un tableau d'enregistrements `etoile` regroupant les informations sur les `N` étoiles, et un autre tableau d'enregistrements `exoplanete` pour stocker les informations sur d'éventuelles exoplanètes détectées grâce aux enregistrements de luminosité des étoiles.

Le tableau d'enregistrements `etoile` contient les champs suivants :

- `ide` : l'identifiant de l'étoile, chaîne de caractères unique ex : `'KP20H3'`
- `type` : type de l'étoile, chaîne de caractères non unique, ex :`'dwarf'`
- `temp` : température de l'étoile, nombre positif
- `radius` : rayon de l'étoile, nombre positif

- `mesuresE` : matrices `2*M`, la première ligne contient les `M` mesures de luminosité en unité de flux, la deuxième ligne contient les temps `t` de mesures en secondes, la première mesure est ramenée au temps `t = 0` . La valeur de `M` varie d'un enregistrement à l'autre.
- `nbrE`: nombre d'exoplanètes suspectées (valeur initialisée à zéro).

Le tableau d'enregistrements `exoplanete` contient les champs suivants :

- `idp` : l'identifiant de l'exoplanète, chaîne de caractères unique.
- `ide` : l'identifiant de l'étoile à laquelle l'exoplanète est associée. C'est une chaîne de caractères non unique dans ce tableau d'enregistrements, car plusieurs exoplanètes peuvent être associées à une même étoile.
- `mesuresP` : matrices `2*M` représentant les mesures de luminosité suspectes qui laissent penser qu'une exoplanète se cache derrière ces mesures. Sur la figure, ce seraient les mesures de luminosité faibles (entre 0 et 8 environ). `M` varie là encore d'un enregistrement à l'autre, par contre, la première mesure de `mesuresP` n'est pas forcément à `t = 0`. En d'autres termes, il s'agit ici du sous-ensemble du champ `mesuresE`, suspectes, associé à l'étoile d'identifiant `ide`.

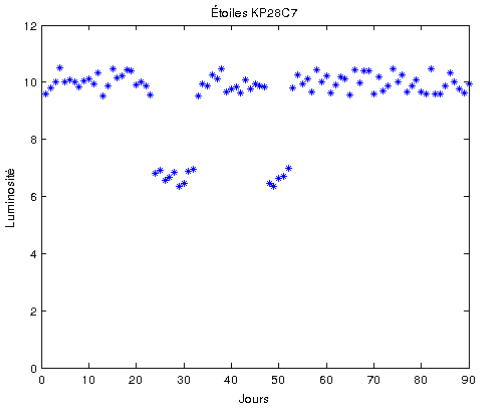
### 4.1 Détection des exoplanètes

Le tableau d'enregistrements `etoile` est donné. Le programme ci-dessous traite ce tableau pour y détecter les mesures suspectes et remplir le tableau d'enregistrements `exoplanete`. Le seuil est calculé par rapport à la moyenne et l'écart type des mesures de luminosité.

```
1     exoplanete = struct([]);
2     for i=1:length(etoile)
3           mtemp = etoile(i).mesuresE;
4           % calcul du seuil
5           moyenne = mean(mtemp(1,:));
6           ecart_type = std(mtemp(1,:));
7           seuil = moyenne - 3*ecart_type;
8           % application du seuil
9           mesures_suspectes = mtemp(:,mtemp(1,:) < seuil);
10          if ~isempty(mesures_suspectes)
11                j = length(exoplanete)+1;
```

```
12                exoplanete(j).idp = [etoile(i).ide '_1'];
13                exoplanete(j).ide = etoile(i).ide;
14                exoplanete(j).mesuresP = mesures_suspectes;
15            end
16    end
```

Le programme précédent suppose qu'il ne peut y avoir qu'une seule série de valeurs suspectes (comme dans la figure au début de l'exercice). Or, plusieurs exoplanètes peuvent tourner autour de l'étoile et créer différentes séries de mesures sous le seuil.




On vous demande durant les deux prochaines questions de modifier le code afin d'enregistrer chacune des séries de mesures suspectes. Chaque série de mesures suspectes est définie par une suite ininterrompue de valeurs sous le seuil. Si la mesure repasse au-dessus du seuil, la série est finie. Si la mesure retourne ensuite sous le seuil, c'est une nouvelle série. Dans la figure ci-contre par exemple, il y a deux séries suspectes.

Chaque série sera enregistrée dans un enregistrement de `exoplanete`. L'identifiant de l'exoplanète suspectée suivra le schéma suivant. Si l'identifiant (`ide`) de l'étoile est `'KP26C7'`, les séries de cette étoile auront pour identifiants `KP26C7_X` où `'X'` dans cette chaîne est le numéro de la série.

4.1.1 Modifier la `ligne 9` pour que `mesures_suspectes` soit un vecteur de booléen de taille égale au nombre de mesures dans `mtemp`, et où chaque valeur indique si oui ou non la mesure est sous le seuil.

mesures_suspectes = mtemp(1, :)<seuil ; (0.5 points)

4.1.2 Modifier la condition de la `ligne 10` pour que le programme rentre dans la structure conditionnelle uniquement s'il existe au moins une valeur booléenne vraie dans `mesures_suspectes`.

if any(mesures_suspectes) (0.5 points)

4.1.3 Modifier les `lignes 11 à 14` pour isoler et enregistrer chaque série de mesures suspectes dans le tableau d'enregistrements `exoplanete`. Vos réponses aux questions a), b) et c) ci-dessous peuvent éventuellement nécessiter plus que 4 lignes de codes pour remplacer les `lignes 11 à 14`.

a) Construire d'abord un ensemble de cellules `series` contenant chaque série de mesures sous le seuil de chaque étoile `i` . Note : on supposera que la première mesure n'est jamais sous le seuil.

(2.5 points)

```
c = 0;                                              (0.5 pts utilisation d'un compteur)
series = {} ;
for j = 2:length(  mesures_suspectes)    (0.5 pts utilisation de for avec indices OK)
                                                    (1 pt, conditions OK pour détecter les séries)
   if mesures_suspectes(j) = = 1 &&  mesures_suspectes(j-1) = =1
      series{c} = [series{c}        mtemp(1,j)];
   elseif mesures_suspectes(j) = = 1 &&  mesures_suspectes(j-1) = =0
      c = c+1;                                      (0.5 pts bon avancement dans series)
      series{c} = [mtemp(1,j)];
   end
end
```

b) Enregistrer chacune de ces séries dans `exoplanete` à l'aide d'une boucle

(1 point)

```
for j = 1:length(series)
   k = length(exoplanete)+1;      (0.5 bon avancement dans exoplanete)
   exoplanete(k).idp = [etoile(i).ide '_' num2str(j)];   (0.5 création étiquette avec num2str(j))
```

```
        exoplanete(k).ide = etoile(i).ide;
        exoplanete(k).mesuresP = series{j};
    end
```

c) Écrire l'instruction pour mettre à jour la valeur du champ `nbrE` dans le tableau d'enregistrements `etoile` :

(0.25 point)

```
etoile(i).nbrE = length(series);
```

### 4.2 Corrélation avec le type d'étoile

Le programme précédent a été exécuté, et vous avez maintenant le tableau d'enregistrements `exoplanete` entièrement rempli avec les exoplanètes suspectées. Les astrophysiciens se demandent maintenant s'il y a un lien entre le type d'une étoile et le nombre d'exoplanètes potentielles autour d'elles. Pour aller plus loin dans leurs investigations, ils aimeraient avoir un programme qui demande à l'utilisateur un type d'étoile (ex : dwarf, giant, etc.), et qui affiche uniquement des informations statistiques sur ce type d'étoile : moyenne du rayon, moyenne de la température, et moyenne/min/max du nombre d'exoplanètes suspectées par étoile.

Exemple d'affichage du programme :

```
>> Entrer un type d'étoile... : dwarf
>> Type dwarf (1547 étoiles de ce type dans la base):
- rayon moyen: 16542.5 km
- nombre moyen d'exoplanètes suspectées: 2.72
- nombre minimal d'exoplanètes suspectées: 0
- nombre maximal d'exoplanètes suspectées: 6
```

Un ingénieur junior a commencé à écrire du code pour vous, mais il est incomplet. Il a laissé des **$X$** partout où il ne savait pas quoi mettre, où X est la lettre de la partie à compléter ci-dessous.

Compléter ces parties, en respectant bien l'exemple d'affichage ci-dessus (précision après la virgule aussi !).

```
1   utype = input('Entrer un type\n','s');
2   cpt = $A$;
3   mR = $B$;
4   nE = [];
5   for i=1:$C$
6       if strcmp($D$)
7           cpt = $E$;
8           mR = mR + etoile(i).radius;
9           nE = $F$;
10      end
11  end
12  if $G$
13      fprintf('Type: %s (%i étoiles de ce type dans la base) \n',utype,cpt);
14      fprintf('- rayon moyen: %.1f km\n',$H$);
15      fprintf('- nombre moyen d''exoplanètes suspectées: $I$\n',mean(nE));
16      fprintf('- nombre minimal d''exoplanètes suspectées: %i\n',min(nE));
17      fprintf('- nombre maximal d''exoplanètes suspectées: %i\n',max(nE));
18  else
19      disp('Aucune étoile ne correspond à ce type')
20  end
```

(2.25 points, 0.25 par réponse)

| | |
|---|---|
| A | 0 |
| B | 0 |
| C | length(etoile) |

| | |
|---|---|
| | |
| D | etoile(i).type,utype |
| E | cpt+1 |
| F | [nE etoile(i).nbrE] |
| G | cpt>0 ou cpt~=0 ou cpt>=1 |
| H | mR/cpt |
| I | %.2f |

# Annexe

| | |
|---|---|
| input() | permet la saisie de données de l'usager. |
| fprintf(), disp(), sprintf() | permettent l'affichage à l'écran des résultats ou des messages. |
| isequal() | vérifie si deux matrices ou chaînes de caractères sont semblables peu importe leur taille. |
| isempty() | vérifie si une matrice est vide. |
| isnumeric() | vérifie si une matrice est numérique |
| isscalar() | vérifie si l'entrée est un scalaire |
| ischar() | vérifie si l'entrée est une chaîne de caractères |
| iscell() | vérifie si l'entrée est un ensemble de cellules |
| lower(), upper() | convertissent une chaîne de caractères en lettres minuscules ou majuscules. |
| rand() | génère une matrice de nombres aléatoires dont les valeurs sont incluses dans l'intervalle ]0, 1[ |
| zeros(), ones() | génèrent une matrice de 0 ou de 1. |
| eye() | génère la matrice identité. |
| size() | donne la taille d'une matrice (le nombre de lignes et le nombre de colonnes) |
| length() | est le maximum de nombre de lignes, nombre de colonnes obtenues avec la fonction size(). |
| floor(), ceil(), fix(), round() | permettent d'arrondir les nombres réels. |
| rem() | donne le reste entier de la division de deux nombres. |
| sum() | calcule la somme des éléments d'un vecteur ou d'une matrice. |
| max(), min() | donnent le plus grand ou le plus petit élément d'un vecteur ou d'une matrice. |
| mean() | calcule la moyenne des éléments d'un vecteur ou d'une matrice. |
| all() | détermine si tous les éléments d'un vecteur/matrice sont différents de 0. |
| any() | détermine si au moins un élément du vecteur/matrice est différent de 0. |
| num2str(), str2num() | convertissent un nombre dans une chaîne de caractères ou une chaîne de caractères dans un nombre. |
| strcmp(), strncmp() | comparent des chaînes de caractères. |
| strfind() | cherche une chaîne de caractères dans une autre chaîne de caractères. |
| char() | crée une chaîne de caractères. |