1) Programme : Répartiteur de tâche (20 points)

Vous avez le contrat d'écrire un programme qui permet de répartir des tâches entre des employés en se basant sur l'estimé du temps requis pour compléter la tâche. Le développement du programme se fera en trois sections, la première implémentera un sous algorithme permettant de trouver l'employé ayant le moins d'heures, la seconde testera une condition logique et la troisième consistera à développer le programme lui-même.

a) fonction : [valeur_min position_min] = trouver_min(tab) (4 points)

Cette fonction implémente un algorithme permettant de trouver l'employé ayant cumulé le moins d'heures jusqu'à maintenant. Attention, vous ne pouvez utiliser la fonction min de Matlab. En fait, il s'agit de l'implémenter.

La fonction reçoit un tableau de nombre d'heures en entrée puis retourne deux paramètres : le nombre d'heures minimal trouvé dans le tableau et l'index de l'élément contenant le moins d'heures.

ex :

```
nb_heures_emp = [35 3 82 33 25];
[nb_heures_min employe_id] = trouver_min(nb_heures_emp)
nb_heures_min
        =
        3
employe_id
        =
        2
```

Aucune validation n'est nécessaire dans cette fonction et vous pouvez supposer que le nombre d'employé sera toujours d'au moins 1.

```
% fonction qui permet de trouver le minimum d'un tableau
function [nb_heures_min employe_id] = trouver_min(nb_heures_emp)

    employe_id = 1;
    nb_heures_min = nb_heures_emp(employe_id);

    for ii=2:numel(nb_heures_emp)
        if nb_heures_emp(ii)<nb_heures_emp(employe_id)
            employe_id = ii;
            nb_heures_min = nb_heures_emp(ii);
        end
    end
```

```
end
```

b) fonction : [sont_superieures] = toutes_valeurs_superieures(tab, n) (2 points)

Cette fonction implémente un algorithme permettant de déterminer si tous les éléments d'un tableau sont supérieurs ou égal à un *n*. Le retour est un état logique, égal à 1, si tous les éléments sont au-dessus de la limite et 0 sinon.

Par exemple, si on applique le test sur le tableau [36 35 42 41 37] avec *n* égal à 35, le retour serait égal à 1.

En contrepartie, si on applique le test sur le tableau [38 30 40 40 43] avec *n* égal à 35, le retour serait égal à 0.

Note : vous ne pouvez utiliser la fonction Matlab all(), pour résoudre ce problème.

```
% fonction qui vérifie que toutes les valeurs d'un tableau sont plus grande ou égal
% à n.
function [sont_superieures] = toutes_valeurs_superieures(tab, n)

sont_superieures = 1;
for ii=1 :numel(tab)
    if tab(ii)<n
        sont_superieures = 0;
    end
end

end
```

c) fonction : programme_attribution_temps (14 points)

Cette fonction implémente le programme complet. Un fichier contenant des commentaires et des déclarations de variables vous est fournis et vous devez le compléter. Les exigences pour le programme sont les suivantes :

- le nombre d'employé entrée par l'usager doit être d'au moins 1, sinon on quitte le programme
- le programme s'exécute tant que l'usager désire attribuer des nouvelles tâches et tant que tous les employés n'ont pas atteint la barre des HEURES_MIN
- Si tous les employés ont accumulé au moins HEURES_MIN de travail, un message s'affiche pour en informer l'usager et le boucle s'interrompt.
- les tâches sont toujours attribuées à l'employé ayant le moins d'heures.
- aucun employé ne peut avoir une charge de travail supérieure à HEURES_MAX
- si une tâche ne peut être attribuée, un message d'information doit s'afficher

```
function nb_heures_emp = programme_attribution_temps()

% définit constantes du programme
HEURE_MIN = 35;
HEURE_MAX = 45;

(2.5)
% demande à l'usager d'entrer le nombre d'employé
nb_employe = input('Entrer le nombre d''employé: ');


% valide la valeur du nombre d'employé ou quitte (retour) le programme
if nb_employe<1
   return;
end



% compteur d'heure de chaque employé
nb_heures_emp = zeros(nb_employe,1);

% réponse de l'usager à savoir s'il désire attribuer une autre tâche
nouvelle_tache = '';
% nombre d'heure de l'employé ayant le moins d'heures
nb_heures_min = 0;
% l'index de l'employé ayant le moins d'heures
employe_id = 0;

% nombre d'heure de la tâche à attribuer
heures_tache = 0;

% condition d'arret du programme atteinte
programme_termine = 0;

% boucle tant que tous les employés n'ont pas au moins 35 heures d'accumulé
while(~programme_termine)

   (1 pts)
   % demande à l'usager s'il veut entrer de nouvelles tâches : oui(o) ou non(n)
```

```
   nouvelle_tache = input('Une nouvelle tâche, oui(o) ou non(n): ',’s’);

   (1.5 pts)
   % s’il désire entrer une nouvelle tâche on procède avec la requête
   if strcmp(nouvelle_tache,'o')

      (1 pts)
      % demande à l'usager d'entrer le nombre d'heure de la tâche
      heures_tache = input('Entrer le nombre d''heure de la tâche: ');

      (1 pts)
      % trouve l'employé ayant le moins d'heure et obtient sa charge courante
      % et son index
      [nb_heures_min employe_id] = trouve_min(nb_heures_emp);


      (2.5 pts)
      % vérifie si l'addition de la tâche entraine le dépassement de la charge de
      % 40 heures. Si oui, affiche un message d’information. Si non, ajoute le
      % temps de la tâche à l’employé.
      if nb_heures_min+heures_tache>HEURE_MAX
          fprintf('La tâche ne peut être attribué');
      else
          nb_heures_emp(employe_id) = nb_heures_emp(employe_id)+heures_tache;
      end


      (1.5 pts)
      % vérifie la condition de fin de programme (si tous les employés ont
      % atteint le plateau minimum, on quitte la boucle)
      if toutes_valeurs_superieures(nb_heures_emp, HEURE_MIN)
         programme_termine = 1;
         fprintf('Tous les employés ont atteint la charge minimale');
      end



   (1 pts)
   % sinon quitte la boucle
   else
      programme_termine = 1;



   end
end

end
```

4) Pré-analyse d'un code secret (10 points)

On vous a assigné la tâche de préparer l'analyse d'un code secret, dont la clé de décryptage est inconnue. L'auteur des messages a néanmoins fait une erreur que vous pouvez exploiter pour faciliter le décodage. Chacun de ses messages commence de la même façon et est suivit par un code identifiant le destinataire.

Votre but est d'écrire deux fonctions, a) une qui identifie le début de chacun des messages et b) une qui permet d'obtenir le code du destinataire.

Tous les messages sont contenus dans un seul tableau. Chaque élément du tableau contient un élément du message et chaque élément est un entier contenu entre 0 et 255. Un message contenant 15 éléments ressemble donc à ceci :

[231 33 233 162 25 72 140 **245 247 41** 248 124 245 205 37]

Tous les messages commencent par la suite de nombre : [245 247 41]. Le début de chaque message est suivi par deux nombres identifiant le destinataire. Ainsi dans le tableau précédent, on trouve que le début du message se fait à la position 8 et l'identifiant du destinataire est [248 124].

a) [debut_msgs] = trouver_messages(code) (5 points)

La première fonction à réaliser en est une qui permet de trouver les positions de début des messages dans un tableau de taille variable. Un appel de cette fonction dans l'exemple précédent aurait retourné 8, mais dans le cas où il y aurait eu plusieurs débuts de message, elle aurait retourné un tableau de plusieurs éléments. Dans le cas où aucun message n'est présent, la fonction doit retourner un tableau vide. (Vous pouvez utiliser le redimensionnement dynamique pour résoudre le problème)

ex : [debut_msgs] = trouver_messages([33 245 247 41 233 12])
debut_msgs
=
2
ex : [debut_msgs] = trouver_messages([33 245 247 41 233 12 245 247 41 89])
debut_msgs
=
[2 7]
ex : [debut_msgs] = trouver_messages([33 245 212 176 233 12 23 251 41])
debut_msgs
=
[]

```
function [debut_msgs] = trouver_messages(code)

DEBUT_MESSAGE = [245 247 41];
debut_msgs = [];

for ii=1:numel(code)-(numel(DEBUT_MESSAGE)-1)

    if code(ii) == DEBUT_MESSAGE(1) &&...
       code(ii+1) == DEBUT_MESSAGE(2) &&...
       code(ii+2) == DEBUT_MESSAGE(3)
        debut_msgs = [debut_msgs ii];
    end

end
```

b) [dest_id] = trouver_destinataire_id(code, pos_message) (5 points)

Cette fonction permet de trouver l'identifiant d'un destinataire à partir de la position de début d'un message. Les identifiants de destinataire sont toujours composé de 2 nombres. Il se peut par contre que le code ait été interrompu avant que l'identifiant du destinataire ait été enregistré au complet. Par exemple, dans le cas où le début du message serait dans les derniers éléments du tableau, l'identifiant du destinataire est inconnu et la fonction retourne un tableau vide.

ex : [dest_id] = trouver_destinataire([33 **245 247 41** 233 12 245 247 41 89], 2)
dest_id
=
[233 12]
ex : [dest_id] = trouver_destinataire([33 245 247 41 233 12 **245 247 41** 89], 7)
dest_id
=
[]

```
function [dest_id] = trouver_destinataire(code, pos_message)

DEST_ID_POS = 3;
dest_id = [];

if (pos_message >= numel(code)-(DEST_ID_POS+1))

   dest_id = zeros(2,1);
```

```
    dest_id(1) = code(pos_message+DEST_ID_POS);
    dest_id(2) = code(pos_message+DEST_ID_POS+1);

end
```