État d'Israël
Ministère de l'Éducation
Type d'examen : א. *Bagrout* pour écoles secondaires
ב. *Bagrout* pour candidats libres
Session de l'examen : été 2015
Numéro de l'examen : 899205, 603
Traduction française (4)

מדינת ישראל
משרד החינוך
סוג הבחינה: א. בגרות לבתי ספר על־יסודיים
ב. בגרות לנבחנים אקסטרניים
מועד הבחינה: קיץ תשע"ה, 2015
מספר השאלון: 899205, 603
תרגום לצרפתית (4)

Informatique ב'

2 unités d'étude (complément pour 5 unités d'étude)

Instructions au candidat

א. Durée de l'examen : trois heures.

ב. Structure du questionnaire et barème de notation :
Ce questionnaire comporte deux parties.

Première partie —	Cette partie comporte quatre questions, répondez à deux d'entre elles.	
	(2 × 25) —	50 points
Deuxième partie —	Cette partie comporte des questions de quatre filières différentes. Répondez uniquement aux questions de la filière dans laquelle vous avez étudié, selon les instructions indiquées dans cette filière.	
	(2 × 25) —	50 points
	Total —	100 points

ג. Matériel auxiliaire autorisé : Tout matériel auxiliaire, à l'exception d'un ordinateur programmable.

ד. Instructions particulières :
(1) Tous les programmes qu'il vous est demandé d'écrire en langage informatique dans la première partie, doivent être écrits dans un seul langage – Java ou C#.
(2) **Inscrivez sur la couverture** de votre cahier d'examen dans quel **langage** vous écrivez – Java ou C#.
(3) **Inscrivez sur la couverture** de votre cahier d'examen le nom de la **filière dans laquelle vous avez étudié**. Cette filière est l'une des quatre suivantes : מבוא לחקר ביצועים, מערכות מחשב ואסמבלי, תכנות מונחה עצמים, מודלים חישוביים.

Remarque : dans les programmes que vous écrivez, vous ne perdrez pas de points si vous écrivez une majuscule au lieu d'une minuscule ou l'inverse.

Tout ce que vous voulez écrire en tant que brouillon (plans, calculs, etc...), doit être écrit dans le cahier d'examen uniquement, sur des pages séparées. Indiquez "טיוטה" en haut de chaque page de brouillon. Écrire des notes de brouillon sur d'autres feuilles que celles du cahier d'examen peut entraîner votre disqualification !

Bonne Chance!

מדעי המחשב ב'

2 יחידות לימוד (השלמה ל-5 יח"ל)

הוראות לנבחן

א. משך הבחינה: שלוש שעות.

ב. מבנה השאלון ומפתח ההערכה:
בשאלון זה שני פרקים.
פרק ראשון — בפרק זה ארבע שאלות, ומהן יש לענות על שתיים.
(25×2) — 50 נקודות
פרק שני — בפרק זה שאלות בארבעה מסלולים שונים. ענה על שאלות רק במסלול שלמדת, לפי ההוראות בקבוצת השאלות במסלול זה.
(25×2) — 50 נקודות
סה"כ — 100 נקודות

ג. חומר עזר מותר בשימוש: כל חומר עזר, חוץ ממחשב הניתן לתכנות.

ד. הוראות מיוחדות:
(1) את כל התכניות שאתה נדרש לכתוב בשפת מחשב בפרק הראשון כתוב בשפה אחת בלבד — Java או C# .
(2) **רשום על הכריכה החיצונית** של המחברת באיזו **שפה** אתה כותב — Java או C# .
(3) **רשום על הכריכה החיצונית** של המחברת את שם ה**מסלול שלמדת**.
המסלול הוא אחד מארבעת המסלולים האלה: מערכות מחשב ואסמבלי, מבוא לחקר ביצועים, מודלים חישוביים, תכנות מונחה עצמים.

הערה: בתכניות שאתה כותב לא יורדו לך נקודות, אם תכתוב אות גדולה במקום אות קטנה או להפך.

כתוב במחברת הבחינה בלבד, בעמודים נפרדים, כל מה שברצונך לכתוב כטיוטה (ראשי פרקים, חישובים וכדומה). רשום "טיוטה" בראש כל עמוד טיוטה.
רישום טיוטות כלשהן על דפים שמחוץ למחברת הבחינה עלול לגרום לפסילת הבחינה!

בהצלחה!

Questions

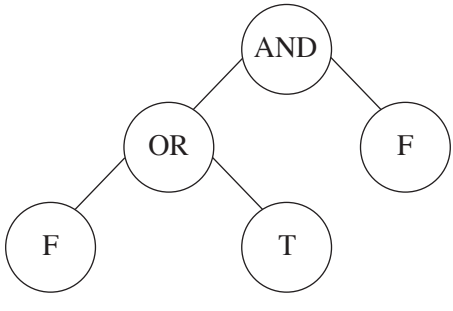
Ce questionnaire comporte deux parties : une première partie et une deuxième partie. Répondez à des questions des deux parties, en suivant les instructions énoncées dans chaque partie.

Première partie (50 points)

Attention : Dans chaque question où une implémentation est requise, vous pouvez utiliser les méthodes des classes Liste, File, Pile, Arbre binaire et Chaîne sans les implémenter. Si vous utilisez des méthodes supplémentaires, implémentez-les.

Répondez à deux des questions 1-4 (25 points par question).

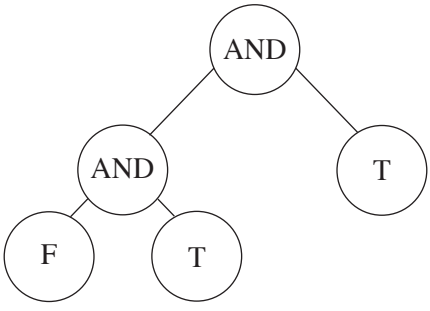
1. Un **arbre d'expression booléenne** est un arbre binaire non vide de type chaîne, représentant une expression booléenne.
Chacune de ses feuilles contient une de ces chaînes : "T" ou "F".
La chaîne "T" représente true , et la chaîne "F" représente false .
Chaque nœud qui n'est pas une feuille contient une de ces chaînes : "AND" ou "OR" .
La chaîne "AND" représente l'opérateur booléen "et", la chaîne "OR" représente l'opérateur booléen "ou".
Chaque nœud qui n'est pas une feuille possède deux fils.
Afin de calculer l'expression booléenne que l'arbre représente, on effectue l'opérateur booléen contenu dans le nœud qui n'est pas une feuille, sur les valeurs obtenues des sous-arbres gauche et droit de ce nœud.
Exemple : l'expression booléenne qui représente l'arbre ci-dessous est ((F OR T) AND (F)) et sa valeur est false.



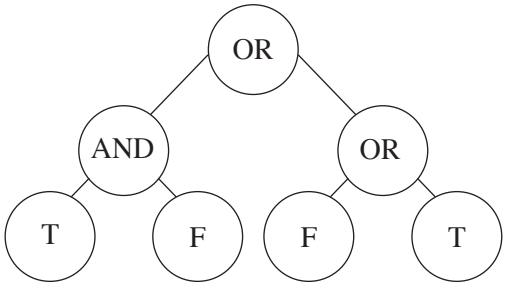
/suite en page 3/

א. Voici trois arbres i-iii , chacun étant un **arbre d'expression booléenne**. Pour chacun de ces arbres, écrivez l'expression booléenne qu'il représente ainsi que la valeur que l'on en obtient.

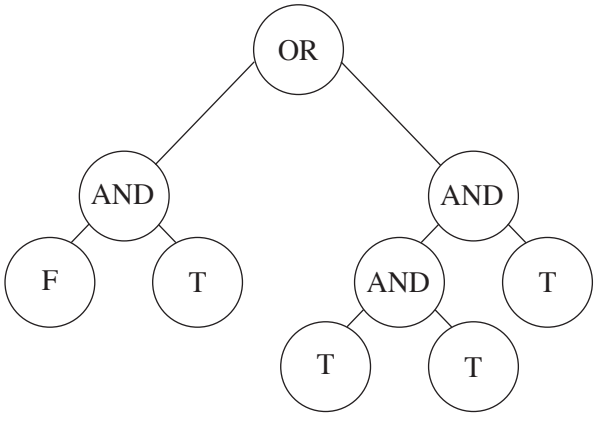
i



ii



iii

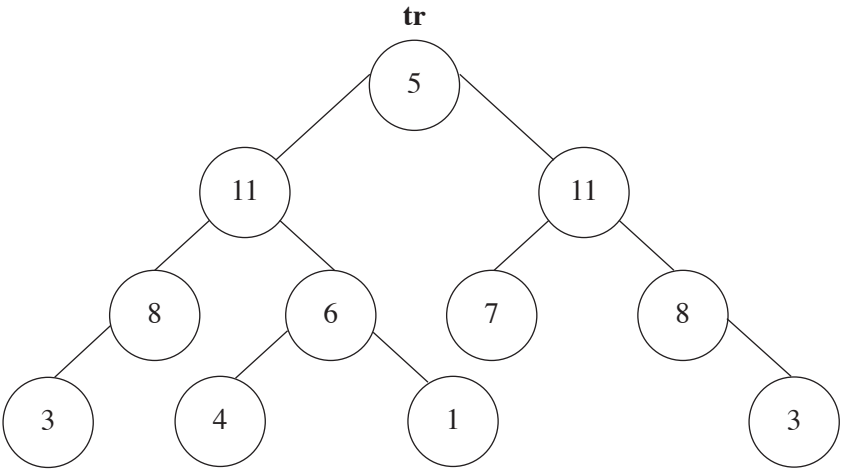


ב. Écrivez en Java ou en C#, une méthode externe qui recevra un **arbre d'expression booléenne**, puis renverra la valeur booléenne (true ou false) de l'expression que l'arbre représente.

/suite en page 4/

2. Cette question comporte deux paragraphes, א-ב, qui ne sont pas liés. Répondez aux deux.

א. Soit l'arbre binaire tr .



Voici la méthode wrap écrite en Java, qui utilise la méthode branch , et la méthode Wrap écrite en C# , qui utilise la méthode Branch .
Attention : La méthode est écrite en deux versions. Répondez au alinéas (1)-(2) ci-dessous d'après la version selon laquelle vous avez étudié.
La version destinée à ceux qui suivent le programme actuel (en Java et en C#) se trouve en page 5,
la version destinée à ceux qui suivent le nouveau programme (en Java et en C#) se trouve en page 6.

(1) Tracez l'exécution de la méthode wrap en Java ou Wrap en C#, pour l'arbre tr donné, puis donnez la valeur renvoyée.
Montrez la trace récursive.

(2) Donnez un exemple d'arbre binaire pour lequel la méthode wrap en Java ou Wrap en C#, renverra true , et un exemple d'arbre binaire pour lequel la méthode renverra false .
Chacun des arbres devra comporter exactement 5 nœuds.

/suite en page 5/

Pour ceux qui suivent le programme actuel

Java

```
public static boolean wrap(BinTreeNode<Integer> tr)
{
    return branch(tr.getLeft(), tr.getRight());
}
public static boolean branch(BinTreeNode<Integer> t1, BinTreeNode<Integer> t2)
{
    if ((t1 == null) && (t2 == null))
        return true;
    if (((t1 != null) && (t2 == null)) || ((t1 == null) && (t2 != null)))
        return false;
    return ((t1.getInfo() == t2.getInfo()) && branch(t1.getLeft(), t2.getRight()));
}
```

C#

```
public static bool Wrap(BinTreeNode<int> tr)
{
    return Branch(tr.GetLeft(), tr.GetRight());
}
public static bool Branch(BinTreeNode<int> t1, BinTreeNode<int> t2)
{
    if ((t1 == null) && (t2 == null))
        return true;
    if (((t1 != null) && (t2 == null)) || ((t1 == null) && (t2 != null)))
        return false;
    return ((t1.GetInfo() == t2.GetInfo()) && Branch(t1.GetLeft(), t2.GetRight()));
}
```

(Attention : le paragraphe ב de la question se trouve en page 7)

/suite en page 6/

Pour ceux qui suivent le nouveau programme

Java

```
public static boolean wrap(BinNode< Integer > tr)
{
    return branch(tr.getLeft(), tr.getRight());
}
public static boolean branch(BinNode<Integer> t1, BinNode<Integer> t2)
{
    if ((t1 == null) && (t2 == null))
        return true;
    if (((t1 != null) && (t2 == null)) || ((t1 == null) && (t2 != null)))
        return false;
    return ((t1.getValue() == t2.getValue()) && branch(t1.getLeft(), t2.getRight()));
}
```

C#

```
public static bool Wrap(BinNode<int> tr)
{
    return Branch(tr.GetLeft(), tr.GetRight());
}
public static bool Branch(BinNode<int> t1, BinNode<int> t2)
{
    if ((t1 == null) && (t2 == null))
        return true;
    if (((t1 != null) && (t2 == null)) || ((t1 == null) && (t2 != null)))
        return false;
    return ((t1.GetValue() == t2.GetValue()) && Branch(t1.GetLeft(), t2.GetRight()));
}
```

(Attention : le paragraphe ב de la question se trouve à la page suivante)

/suite en page 7/

ב. (Il n'y a pas de lien avec le paragraphe א).

Voici une portion de programme écrite en Java et en C#.

Il est donné que a est un tableau de type entier de longueur n .

```
int i = 0;
while (i < n – 1)
{
    if (a[i] > a [i+1])
    {
        a[i]   = a[i] + a[i+1];
        a[i+1] = a[i] – a[i+1];
        a[i]   = a[i] – a[i+1];
        i = 0;
    }
    else
        i++;
}
```

(1) Tracez la portion de programme pour le tableau a de longueur 4 suivant :

a	5	7	8	12

Dans la trace, montrez i et le tableau au terme de chaque itération.

(2) Quelle est la complexité en temps d'exécution de la portion de programme pour un tableau de longueur n , trié par ordre croissant ? Justifiez votre réponse.

(3) Tracez la portion de programme pour le tableau a de longueur 4 suivant :

a	12	8	7	5

Dans la trace, montrez i et le tableau au terme de chaque itération.

(4) Quelle est la complexité en temps d'exécution de la portion de programme pour un tableau de longueur n , trié par ordre décroissant ? Justifiez votre réponse.

/suite en page 8/

3. La classe trajet-d'autobus – **BusRoute** représente le trajet d'une ligne d'autobus. Chaque trajet comporte un nombre quelconque de stations, au minimum deux, dans un ordre déterminé. Chaque station est représentée par deux nombres entiers indiquant son emplacement dans le plan. Chaque station apparaît une fois par trajet. Par exemple, un trajet comportant cinq stations commence à la station (0 , 2) et se termine à la station (5 , 0) :
(0 , 2) → (1 , 4) → (5 , 4) → (3 , 1) → (5 , 0)
L'autobus roule d'une station à l'autre en suivant l'ordre des stations.
La longueur d'un trajet est la somme des distances d'une station à l'autre.
Supposez que soit donnée la classe station – **Station** dont les attributs sont deux nombres entiers x et y représentant l'emplacement de la station dans le plan.
Dans la classe **Station** , ont été définies les deux méthodes :

une méthode constructeur Station(int x , int y)

et une méthode dont la déclaration est :

En Java : double distance(Station other)

En C# : double Distance(Station other)

Cette méthode reçoit une station other et renvoie la distance entre la station actuelle et la station other .

Voici l'interface de la classe trajet-d'autobus – **BusRoute** écrite en Java et en C# :

Java

BusRoute(Station first, Station second)	Méthode constructeur recevant deux stations, puis crée un trajet-d'autobus de deux stations.
void addStation(Station newStation)	Méthode recevant une station puis l'ajoute à la fin du trajet-d'autobus existant. Supposez que cette station n'existe pas dans le trajet.
double routeLength()	Méthode renvoyant la longueur du trajet-d'autobus, c'est à dire la somme des distances d'une station à l'autre.

C#

BusRoute(Station first, Station second)	Méthode constructeur recevant deux stations, puis crée un trajet d'autobus de deux stations.
void AddStation(Station newStation)	Méthode recevant une station puis l'ajoute à la fin du trajet d'autobus existant. Supposez que cette station n'existe pas dans le trajet.
double RouteLength()	Méthode renvoyant la longueur du trajet d'autobus, c'est à dire la somme des distances d'une station à l'autre.

(Attention : les paragraphes de la question se trouvent à la page suivante) /suite en page 9/

א. Écrivez en Java ou C# la déclaration de la classe **BusRoute** ainsi que son attribut/ses attributs. Documentez chaque attribut.

ב. Implémentez en Java ou C# la méthode constructeur pour la classe **BusRoute** .

ג. Implémentez en Java ou C# la méthode ajoutant une station au trajet-d'autobus.

ד. Implémentez en Java ou C# la méthode renvoyant la longueur du trajet-d'autobus.

ה. Écrivez en Java ou C#, dans la méthode principale dans la classe Program , une portion de programme créant le trajet-d'autobus de l'exemple donné au début de la question, puis affichant la longueur de ce trajet-d'autobus.

Remarque : vous pouvez vous servir des méthodes de la classe **Station** sans les implémenter.

/suite en page 10/

4. Dans la société "עבודה יעילה" il existe des tâches : prioritaires, urgentes et normales. Les tâches prioritaires sont effectuées les premières, puis les tâches urgentes, et enfin les tâches normales.

La société dispose d'un système informatisé : **le planificateur de tâches**.

L'insertion de tâches dans **le planificateur de tâches** se fait selon les règles suivantes :

- Une nouvelle tâche prioritaire sera insérée avant toutes les tâches, prioritaires, urgentes et normales, actuellement présentes dans le **planificateur de tâches**.
- Une nouvelle tâche urgente sera insérée après toutes les tâches prioritaires et avant les tâches urgentes et normales actuellement présentes dans le **planificateur de tâches**.
- Une nouvelle tâche normale sera insérée après toutes les tâches, prioritaires, urgentes et normales, actuellement présentes dans le **planificateur de tâches**.

L'extraction d'une tâche à effectuer du **planificateur de tâches** se fait d'après l'ordre constitué dans le **planificateur de tâches**.

Soit la classe **tâche** – **Task** , comportant deux attributs :

content – une chaîne décrivant la tâche,

et code – un nombre entier représentant la tâche : 1 représente une tâche prioritaire; 2 représente une tâche urgente; 3 représente une tâche normale.

Supposez que chaque attribut dispose des méthodes get et set en Java ou Get et Set en C#.

Implémentez la classe **planificateur de tâches** au moyen d'un nombre quelconque de piles et de files, de sorte que la complexité en temps d'exécution des méthodes d'insertion dans le **planificateur de tâches** et des méthodes d'extraction du **planificateur de tâches** soit O(1) .

א. Écrivez en Java ou C# la déclaration de la classe **planificateur de tâches** ainsi que ses attributs. Documentez chaque attribut.

ב. Écrivez en Java ou C# la méthodes constructeur sans paramètres de la classe **planificateur de tâches** .

ג. Écrivez en Java ou C#, dans la classe **planificateur de tâches,** une méthode recevant une **tâche** et l'insérant dans le **planificateur de tâches**, en respectant les règles définies au début de la question.

ד. Écrivez en Java ou C#, dans la classe **planificateur de tâches**, une méthode extrayant la prochaine **tâche** à effectuer, puis qui la renvoie. S'il n'y a pas de **tâche** dans le **planificateur de tâches**, la méthode renverra null .

/suite en page 11/

Deuxième partie (50 points)

Cette partie comporte des questions concernant quatre filières :

מערכות מחשב ואסמבלי, pages 11-14.

מבוא לחקר ביצועים, pages 15-25.

מודלים חישוביים, pages 26-30.

Java–תכנות מונחה עצמים ב, pages 31-40 ; C#–תכנות מונחה עצמים ב, pages 41-50 .

Répondez uniquement aux questions de la filière dans laquelle vous avez étudié.

מערכות מחשב ואסמבלי

Si vous avez étudié dans cette filière, répondez à deux des questions 5-8 (25 points par question).

5. Voici une portion de programme en assembleur.

```
        MOV     BX , 3000H
        MOV     CX , 5
AGAIN:  MOV     AL , CL
        CMP     AL , 3
        JNZ     NEXT            ;(*)
        MOV     [BX] , CL
        INC     BX
NEXT:   LOOPZ   AGAIN
SHOOV:  MOV     AL , CL
        CMP     AL , 3
        JNZ     NEXT1
        MOV     [BX] , CL
        INC     BX
NEXT1:  LOOP    SHOOV           ;(**)
        NOP
```

א. À l'aide d'un tableau de trace, tracez l'exécution de cette portion de programme. Le tableau de trace devra comporter une colonne pour chacun de ces registres : AL , BX , CL .

Répondez aux paragraphes ב-ד ci-dessous. Les paragraphes ne sont pas liés, et chacun d'entre eux traite de la portion de programme ci-dessus.

ב. Déterminez combien de bytes seront inscrits dans la mémoire, en partant de l'adresse 3000H , au terme de l'exécution de la portion de programme.

ג. On a effacé l'instruction signalée par (*). Déterminez ce qui sera inscrit dans la mémoire, en partant de l'adresse 3000H , au terme de l'exécution de la portion de programme sans l'instruction effacée.

ד. Considérez la portion de programme figurant au début de la question, en incluant l'instruction signalée par (*). La ligne signalée par (**) a été remplacée par la ligne NEXT1: LOOPNE SHOOV .
Déterminez ce qui sera inscrit dans la mémoire, en partant de l'adresse 3000H , au terme de l'exécution de la portion de programme après la modification.

/suite en page 12/

6. Cette question comporte deux paragraphes, א-ב, qui ne sont pas liés.
Répondez aux deux.

א. Dans le tableau ci-dessous figurent des instructions constituant des portions de programmes en assembleur.
Recopiez ce tableau dans votre cahier, puis complétez dans chacune des colonnes l'état des drapeaux suite à l'exécution de chacune des instructions.
Supposez qu'avant l'exécution de la portion de programme, les trois drapeaux SF , ZF , CF sont tous initialisés, et que la valeur de BL est –1 .

Instructions	CF	ZF	SF
MOV AL, 3H			
CMP AL, 3H			
CMP AL, 2H			
CMP AL, 5H			
XOR AL, AL			
DEC AL			
MUL BL			

ב. (Il n'y a pas de lien avec le paragraphe א).

(1) Écrivez en assembleur une procédure recevant en tant que paramètre, au moyen de la pile, un nombre entre 0 et 255 .
La procédure stockera dans le registre AX le nombre de "1" qu'il y a dans la représentation binaire de ce nombre.

(2) Soit un tableau ARR comportant 100 éléments dont la taille de chacun est d'un byte. Ce tableau comporte des nombres.
Écrivez en assembleur une portion de programme qui stockera dans le registre CX le nombre d'éléments du tableau qui ont un nombre impaire de "1" dans leur représentation binaire.
Utilisez la procédure que vous avec écrite en ב(1).

/suite en page 13/

7. Cette question comporte deux paragraphes, א-ב, qui ne sont pas liés. Répondez aux deux.

א. Pour chacune des affirmations (1)-(10) ci-dessous, déterminez si elle est vraie ou fausse. Si l'affirmation est fausse, expliquez pourquoi.

(1) Voici une portion de programme en assembleur.

```
PUSH   AL
POP    AH
```

La portion de programme insère dans AH la valeur contenue dans AL .

(2) Avec la méthode du complément à 2 , on peut représenter des nombres entre –128 et +127 au moyen de 8 bits.

(3) L'instruction ci-dessous copie dans BX la valeur de l'adresse dans la mémoire stockée dans AX .

```
MOV    BX , [AX]
```

(4) L'instruction CALL modifie SI .

(5) L'instruction RET augmente la valeur de SP .

(6) L'instruction RET diminue toujours la valeur de IP .

(7) Voici une portion de programme en assembleur dans laquelle la procédure DONOTHING n'effectue rien, et ne modifie pas les valeurs des registres.

```
        MOV     CX , NUMBEROFTIMES
LP:     CALL    DONOTHING
        LOOP    LP
```

Si NUMBEROFTIMES vaut 0 , la procédure DONOTHING sera appelée une fois.

(8) L'instruction ci-dessous initialise BX .

```
XOR    BX , BX
```

(9) La portion de programme ci-dessous initialise DL .

```
MOV    CL , 8
SHR    DL , CL
```

(10) Dans le segment de données ont été définies les variables NUM1 et NUM2 :

```
NUM1   DB  100
NUM2   DB  –156
```

Leur représentation dans la mémoire est identique.

(Attention : le paragraphe ב de la question se trouve à la page suivante) /suite en page 14/

ב. (Il n'y a pas de lien avec le paragraphe א).
Voici une portion de programme écrite en assembleur.

```
MOV   AL , – 1
MOV   DL , 00001111B      ;(*)
MOV   CL , 4
AND   AL , DL
SHR   AL , CL
```

(1) À l'aide d'un tableau de trace, suivez l'exécution de la portion de programme, puis donnez la valeur que prendra AL suite à cette exécution. Le tableau devra comporter une colonne pour chacun de ces registres : AL , CL , DL .

(2) À la ligne signalée par (*), on a remplacé le nombre 00001111B par le nombre 00001111 .
La portion de programme sera-t-elle traduite sans erreur en langage machine ? Justifiez votre réponse.

8. א. Écrivez en assembleur une procédure appelée FINDSECOND qui recevra, au moyen de la pile, deux paramètres : un pointeur sur un tableau de bytes, et le nombre d'éléments dans le tableau.
La procédure FINDSECOND stockera dans le registre AX la deuxième valeur la plus grande du tableau.
Par exemple, pour le tableau :

03H	08H	0CH	01H	09H

La valeur 09H sera stockée dans le registre AX .
Supposez que :
– les nombres dans le tableau sont positifs.
– les nombres dans le tableau sont différents les uns des autres.
– le tableau comporte au moins deux nombres.

ב. Dans le segment de données, on a défini le tableau ARR comportant 100 éléments dont chacun a une taille de un byte. Ce tableau contient des nombres.
Écrivez en assembleur une portion de programme qui affichera le deuxième nombre le plus grand du tableau, dans sa représentation en base 10 .
Supposez que les nombres dans le tableau sont positifs et tous différents.
Utilisez la procédure FINDSECOND que vous avez écrite en א .

/suite en page 15/

מבוא לחקר ביצועים

Si vous avez étudié dans cette filière, répondez à deux des questions 9-12 (25 points par question).

9. Cette question comporte six paragraphes, א-ו, qui ne sont pas liés. Répondez à tous les paragraphes.

א. Soit les contraintes :

$$6x_1 + 3x_2 \leq 18$$
$$-4x_1 + 2x_2 \leq 4$$
$$x_1 + x_2 \geq 3$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$

Dans votre cahier, tracez le domaine des solutions admissibles déterminé selon les contraintes données.
Désignez le domaine des solutions admissibles de ces contraintes, puis inscrivez sur le graphe obtenu les valeurs de x_1 et x_2 pour chacun des sommets du domaine des solutions admissibles.

Chacun des paragraphes ב-ו, se rapporte au problème de programmation linéaire suivant.

Le problème de programmation linéaire :

$$\max\{z = 20x_1 + 15x_2\}$$

soumis aux contraintes suivantes :

$$2x_1 + x_2 \leq 6$$
$$-2x_1 + x_2 \leq 2$$
$$x_1 + x_2 \leq b$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$

b est un paramètre.

Pour $b = 3$, le domaine des solutions admissibles est représenté dans la figure I .

Pour $b = 6$, le domaine des solutions admissibles est représenté dans la figure II .

(Attention : la suite de la question à la page suivante)

/suite en page 16/

Figure II

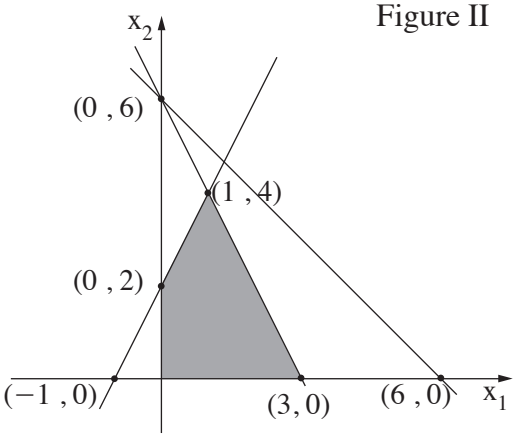
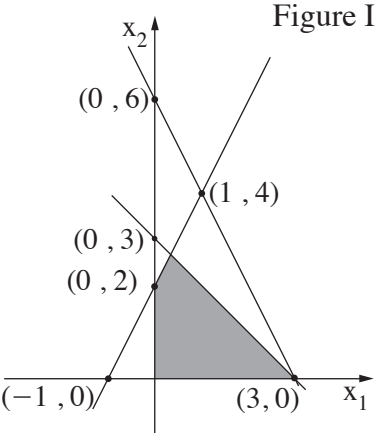


Figure I



ב. Pour $b = 3$, quelle est la solution optimale du problème donné ? Justifiez votre réponse.

ג. Pour $b = 6$, la solution optimale du problème donné est (1 , 4). Expliquez pourquoi.

ד. Trouvez deux valeurs de b (différentes de 6) pour lesquelles la solution optimale du problème donné est (1 , 4). Justifiez votre réponse.

ה. On inverse le signe de l'inéquation de la première contrainte du problème de programmation linéaire précédant le paragraphe ב, ce qui donne à présent : $2x_1 + x_2 \geq 6$.

Pour $b = 3$, tracez le domaine des solutions admissibles.

Quelle est la solution optimale au problème obtenu à la suite de cette modification ? Justifiez votre réponse.

ו. On inverse le signe de l'inéquation des deux premières contraintes du problème de programmation linéaire précédant le paragraphe ב.

Pour $b = 3$, les contraintes sont à présent :

$$2x_1 + x_2 \geq 6$$
$$-2x_1 + x_2 \geq 2$$
$$x_1 + x_2 \leq 3$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$

Existe-t-il une solution optimale au problème obtenu à la suite de cette modification ? Si c'est le cas, déterminez-la, sinon, expliquez pourquoi il n'existe pas de solution.

/suite en page 17/

10. Cette question comporte six paragraphes, א-ו, qui ne sont pas liés.
Répondez à tous les paragraphes.

א. Un problème de transport comporte trois origines et trois destinations.
Les coûts unitaires de chaque origine vers chaque destination sont donnés dans le tableau ci-dessous.

Origines	**Destinations**			**Offre**
	1	2	3	
1	8	9	6	18
2	4	5	8	a
3	5	10	7	10
Demande	12	15	15	

Déterminez quelle doit être l'offre a afin d'obtenir un tableau de coûts et de demandes ne comportant ni destination fictive, ni origine fictive.

ב. Dans le tableau ci-dessous, on donne une partie d'une solution de base admissible à un problème de transport : $x_{12} = 1$, $x_{11} = 9$.

Origines	**Destinations**			**Offre**
	1	2	3	
1	10 (9)	15 (1)	17	10
2	10	18	14	12
3	15	20	18	9
Demande	9	12	10	

Recopiez le tableau dans votre cahier, puis complétez-y la solution de base admissible d'après la méthode du coin nord-ouest.

(Attention : la suite de la question se trouve à la page suivante)

/suite en page 18/

ג. Dans le tableau ci-dessous, on donne une partie d'une solution de base admissible à un problème de transport, et on donne les valeurs de $u_1, u_2, u_3, v_1, v_2, v_3$ correspondant à cette solution.

Origines	**Destinations**			**Offre**	u_i
	1	2	3		
1	2 (20)	5	7	20	2
2	0	8 (10)	4	10	0
3	10	0	8 (10)	15	−8
Demande	20	15	10		
v_j	0	8	16		

Recopiez le tableau dans votre cahier, puis complétez-y la solution de manière à obtenir une solution de base admissible, en considérant les valeurs de $u_1, u_2, u_3, v_1, v_2, v_3$.

(Attention : la suite de la question se trouve à la page suivante)

/suite en page 19/

ד. Dans le tableau suivant, on donne une solution de base admissible à un problème de transport, ainsi que la valeur de u_1 .

Origines	**Destinations**			**Offre**	u_i
	1	2	3		
1	6	7 (15)	9 (3)	18	0
2	2 (10)	0 (0)	6	10	
3	7	12	10 (10)	10	
Demande	10	15	13		
v_j					

(1) Recopiez le tableau dans votre cahier, puis complétez-y les valeurs de u_2 , u_3 , v_1 , v_2 , v_3 .

(2) Expliquez pourquoi la solution donnée n'est pas une solution optimale.

(Attention : la suite de la question se trouve à la page suivante)

/suite en page 20/

ה. Dans le tableau ci-dessous, on donne une solution admissible à un problème de transport, et on donne les valeurs de $u_1, u_2, u_3, v_1, v_2, v_3$ correspondant à cette solution.

Origines	**Destinations**			**Offre**	u_i
	1	2	3		
1	10 (20)	25	30	20	10
2	10 (30)	22	14 (20)	50	10
3	16	20 (40)	20 (20)	60	16
Demande	50	40	40		
v_j	0	4	4		

Choisissez l'affirmation correcte parmi les affirmations 1-4 ci-dessous. Recopiez-la dans votre cahier, puis justifiez votre choix.

1. La solution donnée n'est pas une solution de base admissible.
2. La solution donnée est une solution de base admissible mais non optimale.
3. La solution donnée est une solution optimale unique.
4. La solution donnée est une solution optimale, mais n'est pas une solution optimale unique.

(Attention : la suite de la question se trouve à la page suivante)

/suite en page 21/

ו. Dans le tableau ci-dessous, on donne un problème de transport.
Les données de ce problème, incluant les coûts unitaires depuis chaque origine vers chaque destination, figurent dans ce tableau.

Origines	**Destinations**			**Offre**
	1	2	3	
1	22	30	32	20
2	30	18	24	20
3	15	10	8	30
4	25	15	22	30
Demande	30	40	25	

Une condition est donnée, selon laquelle il ne doit pas rester de stock dans l'origine 1 .
Le problème donné n'est pas équilibré, c'est pourquoi un élève a proposé d'ajouter une destination fictive 4 qui recevra les cinq produits de l'offre excédentaire.
Les coûts unitaires depuis chaque origine vers chaque destination sont donnés dans le tableau ci-dessous.
Les valeurs des coûts de la destination fictive 4 données dans le tableau sont les paramètres x , y , z , w .

Origines	**Destinations**				**Offre**
	1	2	3	4	
1	22	30	32	x	20
2	30	18	24	y	20
3	15	10	8	z	30
4	25	15	22	w	30
Demande	30	40	25	5	

(Attention : la suite du paragraphe se trouve à la page suivante)

/suite en page 22/

Quelle doit être la valeur de chacun des paramètres x , y , z , w afin qu'il soit possible de déterminer une solution de base admissible au problème de transport donné ?

Choisissez la réponse correcte parmi les quatre ci-dessous, recopiez-la dans votre cahier, puis justifiez votre choix.

(M est un très grand nombre).

1. $x = 0$, $y = 0$, $z = 0$, $w = 0$
2. $x = M$, $y = M$, $z = M$, $w = M$
3. $x = M$, $y = 0$, $z = 0$, $w = 0$
4. $x = 0$, $y = M$, $z = M$, $w = M$

/suite en page 23/

11. Cette question comporte deux paragraphes, א-ב, qui ne sont pas liés.
Répondez aux deux.

א. G = (V , E) est un graphe **orienté** représenté par la matrice d'adjacences ci-contre.

	a	**b**	**c**	**d**	**e**
a	0	0	0	0	1
b	0	0	0	1	0
c	1	0	0	0	0
d	0	0	0	0	1
e	0	0	1	0	0

(1) Tracez le graphe G représenté par la matrice d'adjacences.

(2) Trouvez la composante/les composantes fortement connexe(s) (CFC – Strong Connected Components) du graphe donné. Pour chaque CFC que vous avez trouvé, donnez son ensemble de sommets.

(3) Quel est **le nombre minimal** d'arcs qu'il faut ajouter au graphe donné afin que le graphe comporte un seul CFC ? Quel est l'arc ou les arcs qu'il faut ajouter ?

(4) Combien de composantes fortement connexes (CFC) comporte le graphe transposé $G^T = (V , E^T)$ du graphe G donné, et quelles sont-elles ?

Attention – un arc orienté quelconque (a , b) d'un graphe G , se transforme en un arc orienté (b , a) dans le graphe transposé G^T .

ב. (Il n'y a pas de lien avec le paragraphe א).

G = (V , E) est un graphe **non orienté** représenté par la liste d'adjacences ci-contre :

a	→	b	→	c	→	‖	
b	→	a	→	c	→	d	→ ‖
c	→	e	→	a	→	b	→ ‖
d	→	b	→	‖			
e	→	c	→	‖			

(1) Tracez le graphe G représenté par la liste d'adjacences.

(2) Combien de composantes connexes (Connected Components) y a t-il dans le graphe G , et quelles sont-elles ?

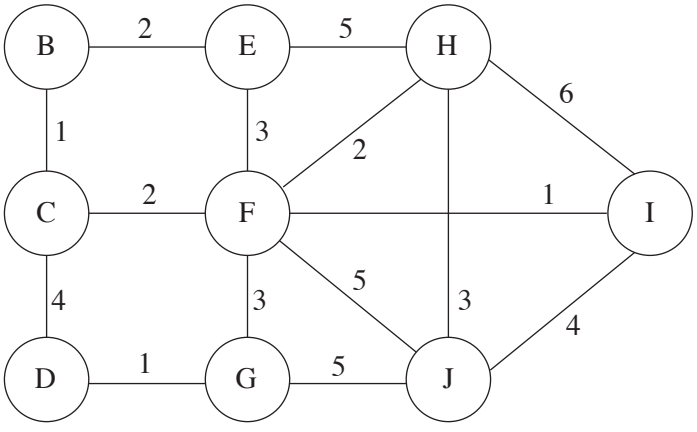
(3) Appliquez un algorithme de recherche en profondeur (DFS) sur le graphe G , en commençant par le sommet a .
Tracez dans votre cahier **uniquement** l'arbre couvrant (DFS) / la forêt couvrante (DFS) obtenu(e).
Basez-vous sur la représentation donnée par la liste d'adjacences.

(4) Appliquez un algorithme de recherche en largeur (BFS) sur le graphe G , en commençant par le sommet a .
Tracez dans votre cahier **uniquement** l'arbre couvrant (BFS) / la forêt couvrante (BFS) obtenu(e).
Basez-vous sur la représentation donnée par la liste d'adjacences.

/suite en page 24/

12. Cette question comporte trois paragraphes, א-ג , qui ne sont pas liés.
Répondez aux trois.

א. Voici un réseau $G = (V , E)$:



(1) Déterminez tous les chemins les plus courts du sommet C au sommet J dans le réseau donné.
Décrivez schématiquement chacun des chemins.

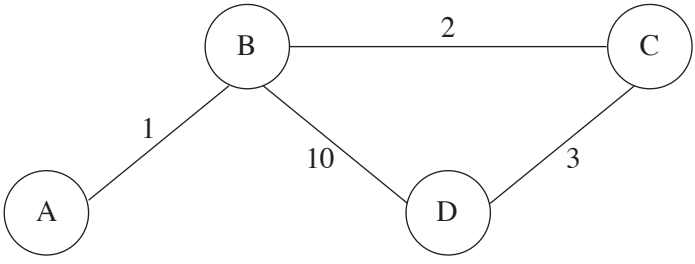
(2) Déterminez l'arbre couvrant minimal selon l'algorithme de Kruskal. Décrivez cet arbre schématiquement.

ב. Soit le graphe **connexe non orienté** $G = (V , E)$ ainsi que la fonction poids $w: E \to R^+$.
On définit une nouvelle fonction poids $cl: E \to R^+$ de cette manière :
Pour chaque arc e de E , $cl(e) = w(e) + a$, a étant un nombre quelconque, supérieur à 0 .
Un élève a affirmé que si T est un arbre couvrant minimal de G selon l'algorithme de Kruskal sous la fonction poids w , alors T est un arbre couvrant minimal de G selon l'algorithme de Kruskal également sous la fonctions poids $c1$.
Cet élève a-t-il raison ? Si c'est le cas, expliquez pourquoi, sinon, donnez un exemple réfutant l'affirmation de l'élève.

(Attention : le paragraphe ג de la question se trouve à la page suivante)

/suite en page 25/

ג. Soit le graphe **connexe non orienté** $G = (V , E)$ ainsi que la fonction poids $w: E \to R^+$.



Pour le graphe G donné, l'arbre des plus courts chemins du sommet A vers le reste des sommets du graphe est :

$A \to B \to C \to D$.

On définit une nouvelle fonction poids $c2: E \to R^+$ de cette manière :

Pour chaque arc e de E , $c2(e) = w(e) + a$, a étant un nombre quelconque, supérieur à 0 .

(1) Donnez l'exemple d'un a pour lequel l'arbre des plus courts chemins du sommet A vers le reste des sommets du graphe ne sera pas différent.

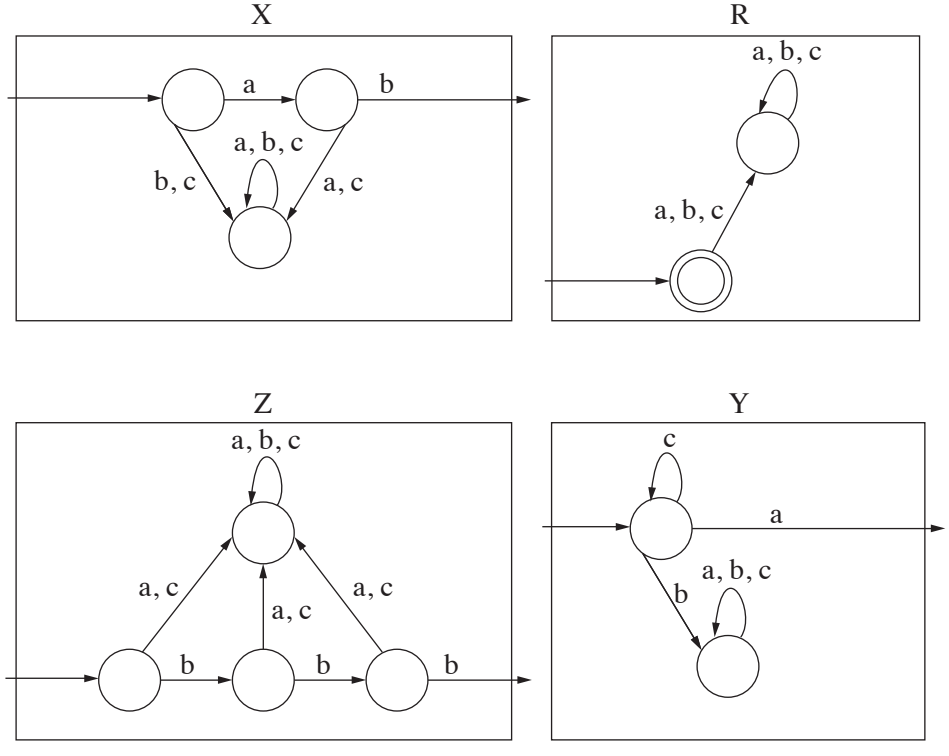
(2) Donnez l'exemple d'un a pour lequel l'arbre des plus courts chemins du sommet A vers le reste des sommets du graphe sera différent.

/suite en page 26/

מודלים חישוביים

Si vous avez étudié dans cette filière, répondez à deux des questions 13-16 (25 points par question).

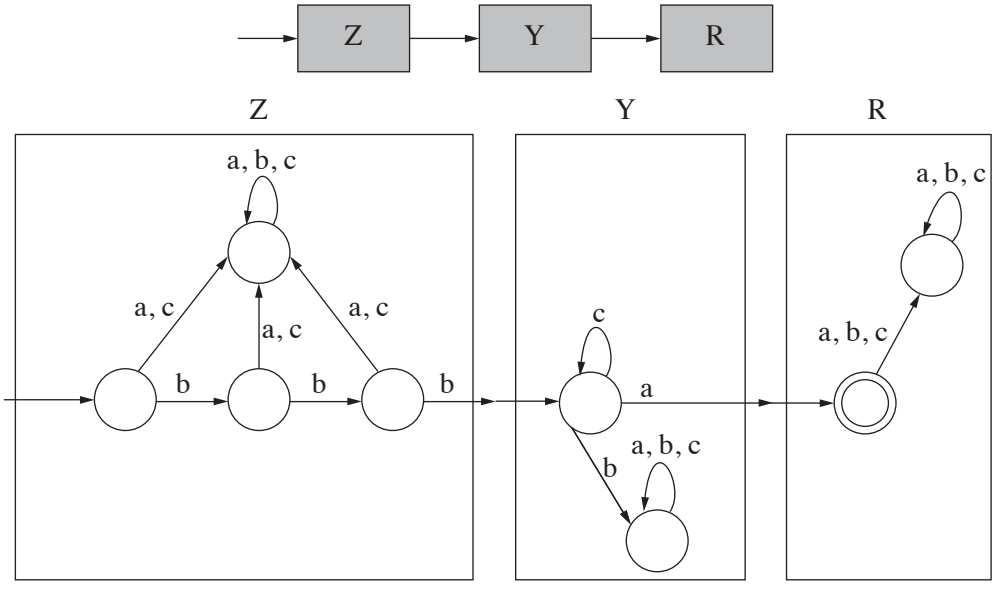
13. Dans l'usine "**אוטומטים בע"מ**", on fabrique les quatre sortes de composants permettant de construire les automates finis déterministes ci-dessous. Chacun des composants n'est pas un automate, mais il est possible de les assembler en un automate fini déterministe, par des liaisons entre eux dans un ordre défini, de gauche à droite. Il est possible d'utiliser plus d'un composant de chaque sorte.



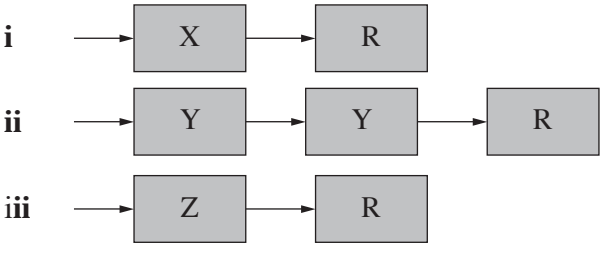
(Attention : la suite de la question se trouve à la page suivante)

/suite en page 27/

Par exemple, afin de construire un automate acceptant le langage $\{b^3 c^n a \mid n \geq 0\}$, il faut relier les composants ci-dessous, dans l'ordre indiqué, de gauche à droite :



א. Écrivez le langage accepté par chacun des automates i-iii ci-dessous.



ב. Soit les langages L_1 , L_2 , L_3 :

$$L_1 = \{ab^4 c^n a \mid n \geq 0\}$$
$$L_2 = \{c^n a^2 b^7 ab \mid n \geq 0\}$$
$$L_3 = \{b^6 c^n a^2 b^4 \mid n \geq 0\}$$

Pour chacun des langages L_1 , L_2 , L_3 , donnez les composants nécessaires parmi les composants X , Y , Z et R de gauche à droite, selon l'ordre dans lequel il faut les relier afin d'obtenir un automate fini déterministe acceptant ce langage.
Il est possible d'utiliser plus d'un composant de chaque sorte.

/suite en page 28/

14. Cette question comporte trois paragraphes, א-ג , qui ne sont pas liés.
Répondez aux trois.

א. Voici le langage L sur l'alphabet {a , b} .

L est l'ensemble des mots dans lesquels la valeur absolue de la différence entre le nombre de fois qu'apparaît le caractère a et le nombre de fois qu'apparaît le caractère b , vaut 1 , et pour chaque antécédent* de chaque mot appartenant au langage L , la valeur absolue de cette différence vaut au maximum 1 .

* L'antécédent d'un mot w , est tout mot obtenu en supprimant un nombre quelconque de caractères à partir de la fin du mot w , en incluant le mot vide et le mot w lui-même.

Exemple : w = abba

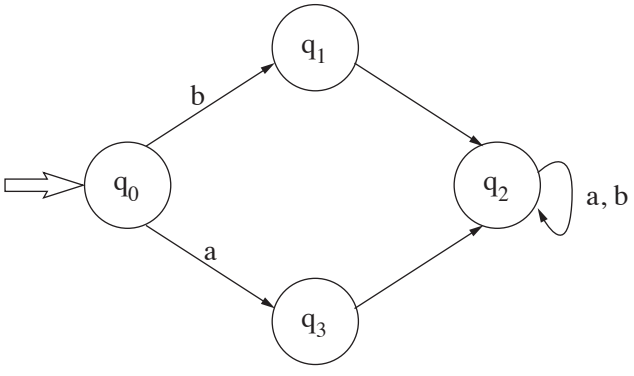
Tous les antécédents du mot w sont : ε , a , ab , abb , abba

Exemple d'un mot appartenant au langage L : ababa

Exemples de mots n'appartenant pas au langage L : aaabb , baab , ε .

Voici le diagramme partiel d'un automate fini déterministe acceptant le langage L .

Il manque à ce diagramme des transitions, des symboles d'entrée et des états acceptants.



Le diagramme comporte tous les états de l'automate.

Recopiez ce diagramme dans votre cahier, puis complétez-le de manière à ce que l'automate accepte le langage L .

Complétez les transitions manquantes et les symboles d'entrée manquants, puis signalez tous les états acceptants.

Attention : il ne faut ni ajouter, ni retirer d'états à l'automate.

(Attention : la suite de la question se trouve à la page suivante) /suite en page 29/

ב. Soit deux langages réguliers L_1 , L_2 .

On définit $L_3 = \{w_1 w_2 \mid w_1 \in L_1 , w_2 \in L_2 , |w_1| = |w_2|\}$.

Donnez l'exemple de deux langages L_1 , L_2 tels que : L_1 , L_2 sont des langages réguliers et L_3 est un langage **non** régulier.

Expliquez votre réponse.

ג. Voici les langages réguliers L_1 , L_2 , L_3 sur l'alphabet $\{a , b , c\}$.

$L_1 = \{a^n b^k \mid n , k \geq 0\}$

$L_2 = \{c^n \mid n \geq 0\}$

$L_3 = \{\varepsilon\}$

Voici le langage L .

$L = \{c^n b^k a^j \mid n > 0 , k , j \geq 0\}$

Montrez que le langage L est régulier, uniquement au moyen des langages L_1 , L_2 , L_3 et de propriétés de fermeture.

Expliquez votre réponse.

15. Voici le langage L .

$$L = \left\{ a^s b^{2s} a^{i_1} b^{j_1} a^{i_2} b^{j_2} \ldots a^{i_n} b^{j_n} \middle| \begin{array}{l} s \geq 1 , n \geq 1 \\ \text{pour tout } k, 1 \leq k \leq n \text{ , se vérifie } i_k \geq 1 , j_k \geq 1 \end{array} \right\}$$

א. Écrivez le mot le plus court du langage L .

ב. Construisez un automate pile acceptant le langage L .

/suite en page 30/

16. Cette question comporte deux paragraphes, א-ב , qui ne sont pas liés.
Répondez aux deux.

א. Voici les langages L_1 et L_2 sur l'alphabet {0,1} :

$$L_1 = \{w \in \{0,1\}^* \mid \#_0(w) \leq \#_1(w) \leq 2 \times \#_0(w)\}$$
$$L_2 = \{0^s \mid s \geq 0\}$$

$\{0,1\}^*$ est l'ensemble de tous les mots sur l'alphabet $\{0,1\}$, en incluant le mot vide,

$\#_0(w)$ indique le nombre de 0 dans le mot w ,

$\#_1(w)$ indique le nombre de 1 dans le mot w .

Soit deux mots dans L_2 : $w_1 = 0^i$ et $w_2 = 0^j$ tels que : $i < j$.

Trouvez un mot w vérifiant :

$$w_1 \cdot w \in L_1$$
$$w_2 \cdot w \notin L_1$$

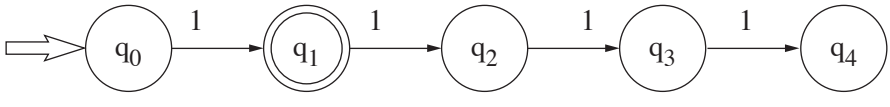
ב. (Il n'y a pas de lien avec le paragraphe א)

Voici le langage L sur l'alphabet {0,1} :

$$L = \{10^i(11)^j \mid i, j \geq 0\} \cup \{10^k 1 \mid k \geq 0\}$$

Voici le diagramme partiel d'un automate fini **non déterministe** acceptant le langage L .

Il manque à ce diagramme des transitions, des symboles d'entrée et des états acceptants.



Le diagramme comporte tous les états de l'automate **non déterministe**.

Recopiez ce diagramme dans votre cahier, puis complétez-le de manière à ce que l'automate accepte le langage L .

Complétez les transitions manquantes et les symboles d'entrée manquants, et signalez les états acceptants manquants.

Attention : il ne faut ni ajouter, ni retirer d'états à l'automate.

/suite en page 31/

תכנות מונחה עצמים

Si vous avez étudié dans cette filière et que vous rédigez en Java, répondez à deux des questions 17-20 (25 points par question).

17. Cette question comporte trois paragraphes, א-ג, qui ne sont pas liés. Répondez aux trois.

א. Voici une portion de code dans la méthode principale de la classe Run :

```
C c = new A();
B b1 = (B) (new A());
B b2 = new D();
A a = new D();
```

(1) Supposez que A , B , C , D sont des classes , et il est donné que chacune d'entre-elles a au maximum une classe héritante. Donnez deux hiérarchies de classes possibles de sorte que la portion de code soit valide.

(2) Il est également donné que la méthode f() est définie uniquement dans la classe B , et il est donné que le programme principal comporte également les instructions suivantes :

```
c.f();
b1.f();
```

Déterminez pour laquelle des deux hiérarchies de classes possibles que vous avez proposées en א (1) ces instructions sont valides. Recopiez cette hiérarchie de classes, puis expliquez pourquoi dans l'autre hiérarchie de classes, les instructions ne sont pas valides.

ב. Voici une portion de programme dans la méthode principale de la classe Program :

```
CC  c = new AA();
BB b1 = (BB) (new AA(2));
```

Parmi AA , BB , CC , déterminez qui ne peut pas être une interface. Expliquez pourquoi.

(Attention : le paragraphe ג de la question se trouve à la page suivante) /suite en page 32/

ג. Voici les classes A , B et la classe Run comportant une méthode principale main . Donnez la sortie de la méthode principale.

```
public class A
{
	public A()
	{
		f();
	}
	public void f()
	{
		System.out.println("Class A");
	}
}

public class B extends A
{
	public B()
	{
		super();
	}
	public void f()
	{
		System.out.println("Class B");
	}
}

public Class Run
{
	public static void main(String[] args)
	{
		A a = new B();
	}
}
```

/suite en page 33/

18. Voici les classes AA , BB , CC , DD et la classe Run comportant une méthode principale.

```
public class AA
{
   protected int i;
   public AA(int i)
   {
        this.i = i;
   }
}

public class BB extends AA
{
   public BB(int i)
   {
        super(i + 1);
   }
   public boolean what(Object other)
   {
        return ((other != null) &&
                (other instanceof BB) &&
                (this.i == ((BB)other).i));
   }
   public boolean what (BB other, int k)
   {
        return ((other != null) &&
                (this.i − k == ((BB)other).i));
   }
}

public class CC extends BB
{
   public CC(int i)
   {
        super(i);
   }
}
```

```
public class DD extends BB
{
   public DD(int i)
   {
        super(i + 1);
   }
   public boolean what(BB other , int k)
   {
        return super.what(other , 1);
   }
}

public class Run
{
   public static void main(String[] args)
   {
        AA a = new AA(1);
        BB b = new BB(1);
        CC c = new CC(1);
        DD d = new DD(1);
        BB b1 = new DD(1);
        AA c1 = new CC(1);
        //***
   }
}
```

(Attention : la suite de la question se trouve à la page suivante)

/suite en page 34/

א. Dessinez la hiérarchie des classes AA , BB , CC , DD .

ב. Tracez la méthode principale.
Lors de la trace, écrivez les valeurs des variables, et pour chaque objet, écrivez les valeurs de ses attributs.

ג. Voici les instructions i-iii .

```
i    a = c;
ii   b = a;
iii  c = (CC) b1;
```

Pour chacune des instructions i-iii :

- Écrivez l'instruction à l'emplacement //*** dans la méthode principale de la classe Run .
- Déterminez si l'instruction est valide ou non.
  Si l'instruction est valide, présentez les modifications apportées aux objets suite à cette instruction.
  Si l'instruction n'est pas valide, justifiez votre réponse puis écrivez si l'erreur est une erreur de compilation ou une erreur d'exécution.

ד. Voici les instructions i-iv .

```
i    System.out.println(b1.what(b));
ii   System.out.println(b1.what(b , 1));
iii  System.out.println(((CC) c1).what(c));
iv   System.out.println(d.what(a));
```

Pour chacune des instructions i-iv :

- Écrivez l'instruction à l'emplacement //*** dans la méthode principale de la classe Run .
- Écrivez ce que sera la sortie suite à chacune des instructions.

/suite en page 35/

19. Voici l'interface IThing , les classes A , B , C , D , et une méthode principale de la classe Run .

א. Tracez la méthode principale, puis écrivez la sortie obtenue.
Lors de la trace, écrivez les valeurs des variables, et pour chaque objet, écrivez les valeurs de ses attributs.

```
public interface IThing
{
    public int value();
}

public class A implements IThing
{
    private int a;
    public A(int a)
    {
        this.a = a;
    }
    public int value()
    {
        return this.a;
    }
}
public class B implements IThing
{
    protected IThing a , b;
    public B(IThing a , IThing b)
    {
        this.a = a;
        this.b = b;
    }
    public int value()
    {
        return this.a.value() + this.b.value();
    }
}
```

(Attention : la suite du paragraphe se trouve à la page suivante)

/suite en page 36/

```
public class C extends B
{
	public C(IThing a, IThing b)
	{
		super(a , b);
	}
	public int value()
	{
		return this.a.value() * this.b.value();
	}
}

public class D implements IThing
{
	private IThing[] things;
	private int  limit;
	public D(IThing[] things , int limit)
	{
		this.things = things;
		this.limit = limit;
	}
	public int value()
	{
		int val = 0;
		for (int i = 0; i < limit; i++)
		{
			val += this.things[i].value();
		}
		return val;
	}
}
```

(Attention : la suite du paragraphe se trouve à la page suivante)

/suite en page 37/

```
public class Run
{
    public static void main(String[] args)
    {
        IThing[] things = new IThing[5];
        things[0] = new A(2);
        things[1] = new B(things[0] , things[0]);
        things[2] = new C(things[0] , things[1]);
        things[3] = things[2];
        things[4] = new D(things , 4);      //    (*)
        for(int i = 0; i < things.length; i++)
        {
            System.out.println(things[i].value());
        }
    }
}
```

ב. Expliquez brièvement quelle sera l'influence sur l'exécution du programme si l'instruction de la ligne signalée par (*) était remplacée par l'instruction suivante :

```
things[4] = new D(things , 5);
```

/suite en page 38/

20. La société "**צעצועים זה הם**" souhaite informatiser la base de données des jouets de son entrepôt principal.

Pour cela, deux classes ont été définies pour deux sortes de jouets :

La classe **Doll** pour les poupées, et la classe **Car** pour les voitures.

La classe **Doll** comporte cinq attributs et deux méthodes.

Les attributs : le nom de la poupée, le prix de base de la poupée, la couleur des cheveux, le nombre d'accessoires joints, le prix d'un accessoire (tous les accessoires ont le même prix).

Les méthodes :

(i) Renvoi du prix client de la poupée. Le prix client de la poupée correspond à son prix de base, plus le prix des accessoires qui lui sont joints (calculé d'après le nombre d'accessoires multiplié par le prix d'un accessoire).

(ii) Actualisation du prix de base d'une poupée par une augmentation de p pourcents. p est un nombre réel reçu en tant que paramètre.

La classe **Car** comporte quatre attributs et deux méthodes.

Les attributs : le nom de la voiture, le prix de base de la voiture, le classement de la taille de la voiture, la couleur de la voiture.

L'attribut "classement de la taille de la voiture" est représenté par un nombre entier : 0 – une petite voiture, 1 – une voiture moyenne, 2 – une grande voiture.

Les méthodes :

(i) Renvoi du prix client de la voiture. Le prix client de la voiture est fixé d'après sa taille.

Le prix client d'une petite voiture correspond à son prix de base,
le prix client d'une voiture moyenne correspond à son prix de base + 15 sh,
et le prix client d'une grande voiture correspond à son prix de base + 30 sh.

(ii) Actualisation du prix de base d'une voiture par une augmentation de p pourcents. p est un nombre réel reçu en tant que paramètre.

(Attention : la suite de la question se trouve à la page suivante)

/suite en page 39/

Voici le diagramme des classes de la base de données informatisée des jouets :

Doll
— String name — double basePrice — String color — int accNums — double accPrice
+ double price() + void updatePrice(double percent) + Doll(String name , double basePrice , String color , int accNum , double accPrice)

Car
— String name — double basePrice — String color — int size
+ double price() + void updatePrice(double percent) + Car (String name , double basePrice , String color , int size)

Légende :
— indique private
+ indique public

(Attention : les paragraphes de la question se trouvent à la page suivante)

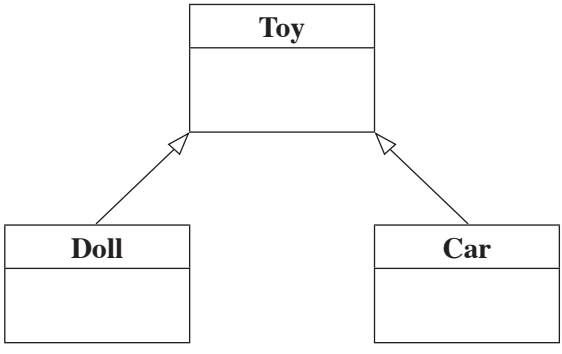
/suite en page 40/

א. On a défini une nouvelle classe **Toy** , dont héritent les classes **Doll** et **Car** .

Voici la structure du diagramme de classes pour les classes **Toy** , **Doll** et **Car** .

Recopiez ce diagramme dans votre cahier puis inscrivez-y, pour chaque classe, les attributs et les déclarations des méthodes, de la manière correspondant au mieux aux principes de la programmation orientée objet (encapsulation, héritage, polymorphisme).

Écrivez une documentation pour chaque attribut.



Dans les paragraphes ב-ג ci-dessous, reportez-vous au diagramme de classes que vous avez écrit en א.

ב. Implémentez en Java la méthode price , dans toutes les classes où elle apparaît.

ג. Écrivez en Java une méthode principale qui effectuera les deux tâches (1) et (2) suivantes

(1) Création d'un objet de type **Doll** , et d'un objet de type **Car** (avec des valeurs de votre choix).

(2) Calcul et affichage de la somme des prix client de ces deux jouets (poupée et voiture).

/suite en page 41/

תכנות מונחה עצמים

Si vous avez étudié dans cette filière et que vous rédigez en C#, répondez à deux des questions 21-24 (25 points par question).

21. Cette question comporte trois paragraphes, א-ג, qui ne sont pas liés. Répondez aux trois.

א. Voici une portion de code dans la méthode principale de la classe Run :

```
C c = new A();
B b1 = (B) (new A());
B b2 = new D();
A a = new D();
```

(1) Supposez que A , B , C , D sont des classes , et il est donné que chacune d'entre-elles a une classe héritante au maximum. Donnez deux hiérarchies de classes possibles de sorte que la portion de code soit valide.

(2) Il est également donné que la méthode F() est définie uniquement dans la classe B , et il est donné que le programme principal comporte également les instructions suivantes :

```
c.F();
b1.F();
```

Déterminez pour laquelle des deux hiérarchies de classes possibles que vous avez proposées en א (1) ces instructions sont valides. Recopiez cette hiérarchie de classes, puis expliquez pourquoi dans l'autre hiérarchie de classes, les instructions ne sont pas valides.

ב. Voici une portion de programme dans la méthode principale de la classe Program :

```
CC  c = new AA();
BB b1 = (BB) (new AA(2));
```

Parmi AA , BB , CC , déterminez qui ne peut pas être une interface. Expliquez pourquoi.

(Attention : le paragraphe ג de la question se trouve à la page suivante) /suite en page 42/

ג. Voici les classes A , B et la classe Run comportant une méthode principale Main . Donnez la sortie de la méthode principale.

```
public class A
{
    public A()
    {
        F();
    }
    public virtual void F()
    {
        Console.WriteLine("Class A");
    }
}

public class B : A
{
    public B(): base()    {   }
    public override void F()
    {
        Console.WriteLine("Class B");
    }
}

public Class Run
{
    public static void Main()
    {
        A a = new B();
    }
}
```

/suite en page 43/

22. Voici les classes AA , BB , CC , DD et la classe Run comportant une méthode principale.

```
public class AA
{
  protected int i;
  public AA(int i)
  {
    this.i = i;
  }
}

public class BB : AA
{
  public BB(int i) : base(i + 1) {  }
  public virtual bool What(Object other)
  {
    return  ((other != null) &&
             (other is BB) &&
             (this.i == ((BB)other).i));
  }
  public virtual bool What (BB other, int k)
  {
    return  ((other != null) &&
             (this.i – k == ((BB)other).i));
  }
}

public class CC : BB
{
  public CC(int i) : base (i)  {  }
}
```

```
public class DD : BB
{
  public DD(int i) : base(i + 1){  }
  public override bool What (BB other ,
int k)
  {
    return base.What(other , 1);
  }
}

public class Run
{
  public static void Main()
  {
    AA a = new AA(1);
    BB b = new BB(1);
    CC c = new CC(1);
    DD d = new DD(1);
    BB b1 = new DD(1);
    AA c1 = new CC(1);
    //***
  }
}
```

(Attention : la suite de la question se trouve à la page suivante)

/suite en page 44/

א. Dessinez la hiérarchie des classes AA , BB , CC , DD .

ב. Tracez la méthode principale.
Lors de la trace, écrivez les valeurs des variables, et pour chaque objet, écrivez les valeurs de ses attributs.

ג. Voici les instructions i-iii .

```
i    a = c;
ii   b = a;
iii  c = (CC) b1;
```

Pour chacune des instructions i-iii :

- Écrivez l'instruction à l'emplacement //*** dans la méthode principale de la classe Run .
- Déterminez si l'instruction est valide ou non.
  Si l'instruction est valide, présentez les modifications apportées aux objets suite à cette instruction.
  Si l'instruction n'est pas valide, justifiez votre réponse puis écrivez si l'erreur est une erreur de compilation ou une erreur d'exécution.

ד. Voici les instructions i-iv .

```
i    Console.WriteLine(b1.What(b));
ii   Console.WriteLine(b1.What(b , 1));
iii  Console.WriteLine(((CC) c1).What(c));
iv   Console.WriteLine(d.What(a));
```

Pour chacune des instructions i-iv :

- Écrivez l'instruction à l'emplacement //*** dans la méthode principale de la classe Run .
- Écrivez ce que sera la sortie suite à chacune des instructions.

/suite en page 45/

23. Voici l'interface IThing , les classes A , B , C , D , et une méthode principale de la classe Run .

א. Tracez la méthode principale, puis écrivez la sortie obtenue.
Lors de la trace, écrivez les valeurs des variables, et pour chaque objet, écrivez les valeurs de ses attributs.

```
public interface IThing
{
    int Value();
}

public class A : IThing
{
    private int a;
    public A(int a)
    {
        this.a = a;
    }
    public int Value()
    {
        return this.a;
    }
}
public class B : IThing
{
    protected IThing a , b;
    public B(IThing a , IThing b)
    {
        this.a = a;
        this.b = b;
    }
    public virtual int Value()
    {
        return this.a.Value() + this.b.Value();
    }
}
```

(Attention : la suite du paragraphe se trouve à la page suivante)

/suite en page 46/

```
public class C : B
{
    public C(IThing a, IThing b) : base(a , b) {    }
    public override int Value()
    {
        return this.a.Value() * this.b.Value();
    }
}

public class D : IThing
{
    private IThing[] things;
    private int  limit;
    public D (IThing[] things , int limit)
    {
        this.things = things;
        this.limit = limit;
    }
    public int Value()
    {
        int val = 0;
        for (int i = 0; i < limit; i++)
        {
            val += this.things[i].Value();
        }
        return val;
    }
}
```

(Attention : la suite du paragraphe se trouve à la page suivante)

/suite en page 47/

```
public class Run
{
    public static void Main()
    {
        IThing[] things = new IThing[5];
        things[0] = new A(2);
        things[1] = new B(things[0] , things[0]);
        things[2] = new C(things[0] , things[1]);
        things[3] = things[2];
        things[4] = new D(things , 4);      //   (*)
        for(int i = 0; i < things.Length; i++)
        {
            Console.WriteLine(things[i].Value());
        }
    }
}
```

ב. Expliquez brièvement quelle sera l'influence sur l'exécution du programme si l'instruction de la ligne signalée par (*) était remplacée par l'instruction suivante :

```
things[4] = new D(things , 5);
```

/suite en page 48/

24. La société "**צעצועים זה הם**" souhaite informatiser la base de données des jouets de son entrepôt principal.

Pour cela, deux classes ont été définies pour deux sortes de jouets :

La classe **Doll** pour les poupées, et la classe **Car** pour les voitures.

La classe **Doll** comporte cinq attributs et deux méthodes.

Les attributs : le nom de la poupée, le prix de base de la poupée, la couleur des cheveux, le nombre d'accessoires joints, le prix d'un accessoire (tous les accessoires ont le même prix).

Les méthodes :

(i) Renvoi du prix client de la poupée. Le prix client de la poupée correspond à son prix de base, plus le prix des accessoires qui lui sont joints (calculé d'après le nombre d'accessoires multiplié par le prix d'un accessoire).

(ii) Actualisation du prix de base d'une poupée par une augmentation de p pourcents. p est un nombre réel reçu en tant que paramètre.

La classe **Car** comporte quatre attributs et deux méthodes.

Les attributs : le nom de la voiture, le prix de base de la voiture, le classement de la taille de la voiture, la couleur de la voiture.

L'attribut "classement de la taille de la voiture" est représenté par un nombre entier : 0 – une petite voiture, 1 – une voiture moyenne, 2 – une grande voiture.

Les méthodes :

(i) Renvoi du prix client de la voiture. Le prix client de la voiture est fixé d'après sa taille.

Le prix client d'une petite voiture correspond à son prix de base,
le prix client d'une voiture moyenne correspond à son prix de base + 15 sh,
et le prix client d'une grande voiture correspond à son prix de base + 30 sh.

(ii) Actualisation du prix de base d'une voiture par une augmentation de p pourcents. p est un nombre réel reçu en tant que paramètre.

(Attention : la suite de la question se trouve à la page suivante)

/suite en page 49/

Voici le diagramme des classes de la base de données informatisée des jouets :

Doll
— string name — double basePrice — string color — int accNums — double accPrice
+ double Price() + void UpdatePrice(double percent) + Doll(string name , double basePrice , string color , int accNum , double accPrice)

Car
— string name — double basePrice — string color — int size
+ double Price() + void UpdatePrice(double percent) + Car (string name , double basePrice , string color , int size)

Légende :
— indique private
+ indique public

(Attention : les paragraphes de la question se trouvent à la page suivante)

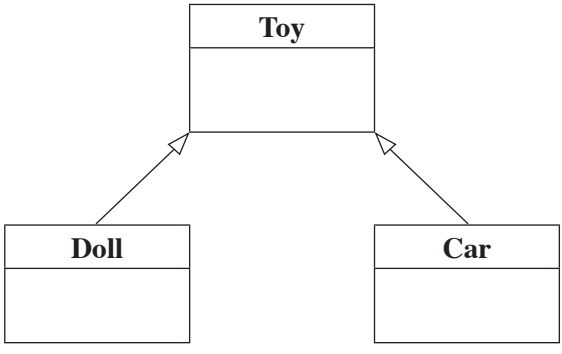
/suite en page 50/

א. On a défini une nouvelle classe **Toy** , dont héritent les classes **Doll** et **Car** .

Voici la structure du diagramme de classes pour les classes **Toy** , **Doll** et **Car** .

Recopiez ce diagramme dans votre cahier puis inscrivez-y, pour chaque classe, les attributs et les déclarations des méthodes, de la manière correspondant au mieux aux principes de la programmation orientée objet (encapsulation, héritage, polymorphisme).

Écrivez une documentation pour chaque attribut.



Dans les paragraphes ב-ג ci-dessous, reportez-vous au diagramme de classes que vous avez écrit en א.

ב. Implémentez en C# la méthode Price , dans toutes les classes où elle apparaît.

ג. Écrivez en C# une méthode principale qui effectuera les deux tâches (1) et (2) suivantes

(1) Création d'un objet de type **Doll** , et d'un objet de type **Car** (avec des valeurs de votre choix).

(2) Calcul et affichage de la somme des prix client de ces deux jouets (poupée et voiture).

Bonne chance!

בהצלחה!