DÉPARTEMENT D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE

SIGLE DU COURS: IFT 1010 (H2000)

NOM DU PROFESSEUR: Nadia El-Mabrouk

TITRE DU COURS: Programmation I

EXAMEN INTRA

Date	: 24 fevrier 2000
Heure	: 8:30-10:30
Salles	: 1360 et Z-250

DIRECTIVES PÉDAGOGIQUES:

- Toute documentation est permise.
- **Inscrivez tout de suite votre nom et code permanent dans la case** (Figure 1).
- Répondez **sur le questionnaire** en utilisant l'espace libre qui suit chaque question.

1. ....
2. ....
3. ....
4. ....
5. ....
6. ....
7. ....

Total ....

Figure 1: Inscrivez votre nom et votre code permanent ici

Question 1 (20 points):

Le programme suivant compte le nombre de réussites et d'échecs. Le chiffre (1) indique une réussite et (0) un échec. Ce programme contient trois erreurs de compilation.

```
import java.io.*;

public class Analyse_faux
{
    private final static int NB_ETUDIANTS = 10;

    public static void main (String[] args) throws IOException
    {
        BufferedReader stdin = new BufferedReader
            (new InputStreamReader(System.in));
        int passages = 0, echecs = 0;
        int etudiant, resultat;
        while (etudiant <= NB_ETUDIANTS)
            {
                System.out.print("Entrer le resultat (1=passage, 0=echec): ");
                resultat = stdin.readLine();
                if ( resultat = 1)
                    passages+=1;
                else
                    echecs+=1;
                ++etudiant;
            }
        System.out.println("Nombre de passages: " + passages);
        System.out.println("Nombre d'echecs: " + echecs);
    }
}
```

i) Trouver les trois erreurs: pour chacune d'elle, la souligner dans le listing, et donner ci-dessous la forme correcte et une explication courte.

ii) Modifier le programme précédent en utilisant une boucle **for** plutôt que la boucle **while** (ne réecrire que la boucle).

Question 2 (12 points):

Chacun des programmes suivants présente une erreur d'exécution ou une erreur logique. Expliquer le problème et dire comment le résoudre en une phrase courte.

(a)
```
for (index=100, index>=1; index++)
    System.out.println(index);
```

(b) Le programme suivant doit afficher les valeurs de 1 à 10:
```
n=1;
while ( n <= 10 )
    System.out.println( ++n );
```

(c) Le programme suivant doit afficher les nombres pairs de 2 à 100 (2 et 100 compris):
```
compteur = 2;
do  {
        System.out.println(compteur);
        compteur+=2;
    }
while (compteur < 100);
```

Question 3 (9 points):

Supposons que l'on ait les déclarations et les initialisations suivantes:

```
int resultat = 10, num=0, num1=3, num2=5;
double d=4.0;
boolean test;
char c1='r', c2;
```

Remplire le tableau suivant avec la valeur correspondante de la variable. Prendre chaque ligne indépendamment, i.e. considérer, pour chaque ligne, les valeurs d'initialisation des variables (ne pas tenir compte des modifications apportées aux valeurs des variables aux lignes précédentes).

Instruction	**Variable**	**Valeur**
resultat = num1 + num2 / 2;	resultat	
resultat = num++;	resultat	
resultat %= 4;	resultat	
resultat = (num1 > num2) ? num1 : num2;	resultat	
d = (double) (num1/num2);	d	
d = num1 / d - (double) (num2) / 2;	d	
test = (resultat > 0 && num1 + num2 > resultat);	test	
c1 += 4;	c1	
c2 = (char) ((c1 >= 'A' && c1 <= 'Z') ? c1 : (c1 - 'a' + 'A'));	c2	

Question 4 (15 points):

On considère le programme suivant:

```
index= 1;
while (index <= 20)
    {
        System.out.print(index);

        if (index % 5 == 0)
            System.out.println();
        else
            System.out.print("  ");
        index++;
    }
```

(a) Quelle est la sortie de ce programme (qu'est-ce qu'il affiche)?

(b) Modifier le programme précédent en remplaçant la boucle **while** par une boucle **do-while**.

(c) Réécrire le programme de façon plus compactée, en utilisant une boucle **for** plutôt que **while**, et en utilisant deux commandes d'affichage plutôt que trois.

Question 5 (14 points):

On considère le programme suivant:

```
for (l=1; l<=8; l++)
     System.out.print('*');
System.out.println();

for (h = 2; h < 4; h++)
    {
        for (l = 1; l <= 8; l++)
            {
                if ( (l==1) || (l==8) )
                    System.out.print('*');

                else
                    System.out.print(' ');
            }
        System.out.println();
    }

for (l=1; l<=8; l++)
     System.out.print('*');
System.out.println();
```

(a) Quelle est la sortie de ce programme (qu'est-ce qu'il affiche)?

(b) Réecrire le programme précédent en supprimant le for en debut et en fin de programme. Pour cela, remplacer les espaces dans le code ci-dessous de telle sorte à obtenir la même sortie que le programme d'origine.

```
for (h =    ; h <=     ; h++)
    {
       for (l =    ; l <=    ; l++)
          if(                                                          )

             System.out.print(' ');
          else
             System.out.print('*');
         System.out.println();
    }
```

Question 6 (15 points):

Écrire un programme qui affiche les sommes successives des termes de la suite:

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} \cdots$$

Autrement dit, le programme affiche les valeurs successives des expressions:

$$1\ ,\ \ 1 - \frac{1}{2}\ ,\ \ 1 - \frac{1}{2} + \frac{1}{3}\ ,\ \ 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4}\ , \cdots$$

L'affichage s'arrête dès que $|\mathbf{somme}-\mathbf{sommePrec}| \leq \mathbf{0.0001}$, où "somme" est la somme à une étape i, et "sommePrec" est la somme à l'étape $i-1$.

On rappelle que la valeur absolue d'un nombre x est obtenue par la méthode **Math.abs(x)**.

Question 7 (15 points):

Le triangle de Pascal à l'ordre N permet de retrouver tous les coefficients binômiaux $\binom{n}{p}$, pour $0 \leq p \leq n \leq N$. Par exemple, le triangle de Pascal à l'ordre 5 est le contenu du tableau:

	0	1	2	3	4	5
0	1					
1	1	1				
2	1	2	1			
3	1	3	3	1		
4	1	$\underline{4}$	$\underline{6}$	4	1	
5	1	5	**10**	10	5	1

Pour ce qui suit, il n'est pas nécessaire de comprendre le sens de ce triangle (comment retrouver les coefficients binômiaux). On s'intéresse uniquement à la façon de le construire. On suppose que les cases vides du tableau (au dessus de la diagonale) contiennent des 0.

La première colonne (colonne n^o 0) du triangle de Pascal ne contient que des 1. Pour chaque case (i, j) du triangle (i^{eme} ligne, j^{eme} colonne), la valeur de cette case est la somme des valeurs des cases $(i-1, j-1)$ et $(i-1, j)$ (case juste en haut à gauche + case juste en haut). Par exemple, la case $(5, 2)$ contenant la valeur 10 (en gras dans le triangle) est obtenue en effectuant la somme des deux valeurs 6 et 4 (soulignées dans le triangle). Remarquer que le calcul d'une ligne i du triangle ne nécessite de connaître que la ligne qui la précède (i.e. la ligne $i - 1$).

On demande d'écrire un programme qui affiche le triangle de Pascal à l'ordre 5 (i.e. exactement le triangle contenu dans le tableau précédent, mais sans les indices des lignes et des colonnes). Pour cela, on utilisera deux tableaux de taille 6: un tableau `lignePrec` contenant la ligne précédente, et un tableau `ligne` contenant la ligne courante. Au départ les deux tableaux sont initialisés à $(1, 0, 0, 0, 0, 0)$. La valeur d'une case $p > 0$ du tableau `ligne` est obtenue par l'instruction:

```
ligne[p]=lignePrec[p-1]+lignePrec[p];
```

Rappelons que:

- La numérotation des cases d'un tableau commence à 0;
- Pour afficher (ou initialiser ou copier) un tableau, il faut afficher (ou initialiser ou copier) chaque case du tableau, l'une après l'autre.

Écrire un programme qui affiche le triangle de Pascal. On vous donne ci-dessous le début du programme, contenant la déclaration des deux tableaux dont vous aurez besoin.

```
// Affichage du triangle de Pascal

public class TrianglePascal
{
    private final static int TAILLE = 6;

    public static void main (String[] args)
    {
        int[] lignePrec = new int[TAILLE], ligne = new int[TAILLE];
```