```
/**
Ce document est le 1er TD pour le cours INF1005C. Il consiste de la résolution de 7
petits problèmes en utilisant du pseudo-code.

Auteur: Johnatan Gao
Date: 2019-09-07
*/



/**
Le premier programme va, tout d'abord, afficher un message qui va demander à
l'utilisateur d'entrer un voltage ("Entrer un voltage V (en volts)"). Ensuite, il va
lire cette donnée et le sauvegarder dans une variable nommée voltage. Suite à cela, le
proramme va afficher un autre message qui va demander à l'utilisateur de entrer une
résistance ("Entrer une résistance R qui n'est pas égale à 0 (en ohms)", pour ensuite
le lire et le sauvegarder dans une variable nommée resistance. Finalement, le programme
va afficher la puissance dissipée en watts en faisant voltage*voltage/resistance("La
puissance dissipée est", voltage*voltage/resistance, "watts").
*/

    AFFICHER "Entrer un voltage V (en volts)"
    LIRE voltage
    AFFICHER "Entrer une résistance R qui n'est pas égale à 0(ohms)"
    LIRE resistance
    AFFICHER "La puissance dissipée est", voltage*voltage/resistance, "watts"



--------------------------------------------------------------------------------

/**
Le deuxième programme va, tout d'abord, afficher un message demandant à l'utilisateur
d'entrer un entier n strictement positif ("Entrer un entier n strictement positif").
Ensuite, il va lire cette donnée et le sauvegarder dans une variable nommée
nombreElements. Suite à cette lecture, TANT QUE nombreElements n'est pas strictement
positif, le programme va afficher "Vous devez rentrer un nombre strictement positif"
demandant à l'utilisateur d'entrer à nouveau cette donnée (après l'avoir lu bien sûr).
Suite à cela, le programme va demander (nombreElements) de fois à l'utilisateur
d'entrer un entier (pour le lire et le sauvegarder dans une variable nommmée
nombreEntree) en incrémentant une variable indiceElement tant qu'elle est inférieur à
nombreElements. À chaque itération, SI nombreEntree est plus grand que nombreAvant
(initialisé à 0), ALORS il va incrémenter cptLongueurTmp (initialisé à 0) de 1, SINON
SI cptLongueurTmp est supérieur à cptLongueur ALORS le programme va affecter la valeur
de cptLongueur par la valeur de cptLongueurTmp. Il va aussi réinitialiser la valeur de
cptLongueurTmp à 0. Finalement, le programme va afficher cptLongueur.
*/

    AFFICHER "Entrer un entier n strictement positif"
    LIRE nombreElements

    TANT QUE nombreElements < 0 FAIRE
        AFFICHER "Vous devez rentrer un nombre strictement positif"
        LIRE nombreElements

    indiceElement = 0
    cptLongueur = 0
    cptLongueurTmp = 0
    nombreAvant = 0



    TANT QUE indiceElement < nombreElements FAIRE

        AFFICHER "Entrer un nombre faisant partie de la suite de taille ", nombreElements
        LIRE nombreEntree

        SI indiceElement == 0 ou nombreEntree > nombreAvant ALORS //La raison pourquoi
        j'ai ajouté la condition indiceElement == 0 est pour au cas où le premier
```

```
            entier de la suite est négative
48              cptLongueurTmp += 1
49          SINON
50              SI cptLongueurTmp > cptLongueur ALORS
51                  cptLongueur = cptLongueurTmp
52
53              cptLongueurTmp = 1
54
55
56          nombreAvant = nombreEntree
57          indiceElement += 1
58
59
60      AFFICHER "La longueur de la plus longue suite croissant est ", cptLongueur
61
62  ----------------------------------------------------------------------------------------
63
64  /**
65  Le troisième programme va, tout d'abord, demander à l'utilisateur d'entrer un mot en
    affichant "Entrer un mot". Il va ensuite lire et sauvegarder cette entrée dans une
    variable nommée mot. Ensuite, le programme va demander à l'utilisateur d'entrer une
    lettre en affichant "Entrer une lettre", puis il va sauvegarder cette entrée dans une
    variable nommmée lettre. Comme dernière entrée, le programme va demander à
    l'utilisateur de entrer un nombre n en affichant "Entrer un nombre n" pour le lire et
    le sauvegarder dans une variable nommmée position. Ensuite, TANT QUE position est
    inférieur à la longueur du mot, il va itérer à travers chaque lettre du mot (en
    utilisant la variable position) et SI la lettre à cette position correspond à lettre
    (variable) ALORS il va incrémenter la variable nombreLettre (initialisé à 0) de 1.
    Finalement, le programme va afficher nombreLettre.
66  */
67
68      AFFICHER "Entrer un mot"
69      LIRE mot
70      AFFICHER "Entrer une lettre"
71      LIRE lettre
72      AFFICHER "Entrer un nombre n"
73      LIRE position
74
75      nombreLettre = 0
76
77      TANT QUE position < longueur de mot FAIRE
78
79          SI mot[position] == lettre ALORS
80              nombreLettre += 1
81
82          position += 1
83
84      AFFICHER nombreLettre
85
86  ----------------------------------------------------------------------------------------
87
88  /**
89  Le quatrième programme va demander à l'utilisateur d'entrer deux valeurs en affichant,
    en premier, "Entrer un nombre n" et deuxièmement "Entrer un nombre k". Le programme va
    ensuite lire les deux entrées et les sauvegarder dans une variable n et k
    respectivement. Par la suite, TANT QUE k est négatif ou supérieur à n, le programme va
    redemander à l'utilisateur d'entrer une valeur de k. Ensuite, elle va afficher le
    résultat de la FONCTION bin(n,k). La FONCTION bin(n,k) consiste d'une boucle qui
    continuer TANT QUE k < n. Dans la boucle, il multiplie nombreBin par (k+1)/(n-k) et il
    incrémente k de 1. À la fin, la fonction retourne nombreBin comme RESULTAT.
90  */
91
92      FONCTION bin(n,k)
93
94          nombreBin = 1
95
96          TANT QUE k < n FAIRE
97
98              nombreBin *= (k+1)/(n-k)
```

```
            k += 1

        RESULTAT nombreBin


    //Cette partie représente le programme principal (main)
    AFFICHER "Entrer un nombre n (le degré du polynôme)"
    LIRE n
    AFFICHER "Entrer un nombre k (le coefficient)"
    LIRE k

    TANT QUE k < 0 ou k > n FAIRE
        AFFICHER "Entrer un nombre k (le coefficient) >= 0 et <=n"
        LIRE k

    AFFICHER bin(n,k) //Ceci est une appel a une fonction


-------------------------------------------------------------------------------------

/**
Le cinquième programme va demander à l'utilisateur d'entrer une phrase en affichant
"Entrer une phrase" et il va lire cette entrée, puis il va le sauvegarder dans une
variable nommée texte. Ensuite, TANT QUE la variable positionTexte (initialisé à 0) + 2
est inférieur à la longueur de texte, SI les trois lettres dans la variable texte à
partir de positionTexte correspondent à 'I', 'N' et 'F', il affiche la variable
positionTexte et termine la boucle. SINON, SI positionTexte est égale à longueur de
texte -2, ALORS le programme va afficher "Ne s'y trouve pas". Il incrémente
positionTexte de 1.
*/

    AFFICHER "Entrer une phrase"
    LIRE texte

    positionTexte = 0

    TANT QUE positionTexte + 2 < longueur de texte FAIRE

        SI texte[positionTexte] == 'I' et texte[positionTexte + 1] == 'N' et
        texte[positionTexte + 2] == 'F' ALORS

            AFFICHER "La position de INF est ", positionTexte
            positionTexte = longueur de texte //Puisque le break n'est pas permi (dans
            le but de terminer la boucle)

        SINON

            SI positionTexte >=  longueur de texte - 2 ALORS
                AFFICHER "Ne s'y trouve pas"

            positionTexte += 1



-------------------------------------------------------------------------------------


/**
Le sixième programme demande à l'utilisateur d'entrer deux données en affichant "Entrer
deux nombres n1 et n2" et il va les lire et sauvegarder dans une variable n1 et n2
respectivement. Il ensuite afficher le résultat de la fonction ppcm(n1,n2) en affichant
"Le ppcm de ", n1 ,"et ", n2 ," est ", ppcm(n1,n2). La fonction pppcm(n1,n2) consiste
d'une boucle TANT QUE qui va continuer tant que la variable ppcmTrouve est fausse. Dans
la boucle, SI le modulo entre la variable multiple(initialisé à 1) multiplié avec la
variable n1 ainsi que la variable n2 est égale à 0, ALORS, la variable ppcmTrouve
devient vrai, SINON il incrémente le multiple de 1. En sortant de la boucle, la
fonction retourne comme RESULTAT multiple*n1.
*/

```

```
151       FONCTION ppcm(n1, n2)
152
153           ppcmTrouve = faux
154           multiple = 1
155
156           TANT QUE ppcmTrouve == faux FAIRE
157
158               SI (n1*multiple) % n2 == 0 ALORS
159                   ppcmTrouve = vrai //Normalement, c'est inutile, car le RESULTAT va
                      terminer la fonction (just in case)
160               SINON
161                   multiple += 1
162
163
164           RESULTAT n1 * multiple
165
166
167       //Cette partie représente le programme principal (main)
168       AFFICHER "Entrer deux nombres n1 et n2"
169       LIRE n1
170       LIRE n2
171
172       AFFICHER "Le ppcm de ", n1 ,"et ", n2 ," est ", ppcm(n1,n2)
173
174   ------------------------------------------------------------------------------------
175
176   /**
177   Le septième programme demande à l'utilisateur d'entrer une donnée en affichant "Entrer
      la hauteur du triangle" et ensuite, il va lire et sauvegarder cette donnée dans une
      variable nommmée hauteur. Ensuite, il va avoir une boucle qui va continuer TANT QUE la
      variable positionLigne (initialisé à 1) est plus petit ou égale à la variable hauteur.
      Dans cette boucle, il va avoir deux boucles: une pour afficher les espaces et l'autre
      pour afficher les étoiles. Pour la boucle des espaces, elle va continuer TANT QUE la
      variable indice (initialisé à 0) est inférieur ou égale à hauteur - positionLigne tout
      en affichant " " et en incrémentant indice. Pour la boucle des étoiles, elle va
      continuer TANT QUE indice (réinitialisé à 0) est inférieur à 2*positionLigne-1 tout en
      affichant "*" et en incrémentant indice. À la fin des deux boucle, le programme va
      sauter de ligne (fin de ligne) et il va incrémenter positionLigne de 1.
178   */
179
180
181       AFFICHER "Entrer la hauteur du triangle"
182       Lire hauteur
183
184       positionLigne = 1
185
186       TANT QUE positionLigne <= hauteur FAIRE
187
188           indice = 0
189
190           TANT QUE indice < hauteur-positionLigne FAIRE
191               AFFICHER " "
192               indice += 1
193           FIN TANT QUE
194
195           indice = 0
196
197           TANT QUE indice < 2*positionLigne-1 FAIRE
198
199               AFFICHER "*"
200               indice += 1
201
202           FIN TANT QUE
203
204           AFFICHER fin de ligne
205
206           positionLigne +=1
207
208       FIN TANT QUE
```