Parmi ces opérations, lesquelles sont spécifiques à une procédure, et ne doivent pas être réalisées par une fonction ?

A. affecter une valeur à une cellule
B. affecter une valeur à une variable
C. retourner une valeur
D. afficher un message à l'utilisateur (`MsgBox`, `Input`, …)

La fonction suivante peut-elle tourner indéfiniment ? Si oui, trouver un exemple, sinon, argumenter.

```
Function Mystère(X as Integer) As Integer
  Mystère = 4
  For i = 1 to (X * X + 1)
    If Mystère > 0 Then
      Mystère = - Mystère * X
    Else
      Mystère = X * X + 1
    End If
  Next i
End Function
```

Combien de tours de boucle effectue l'appel `=Mystère(-3)` ?

```
Function Mystère(X as Integer) As Integer
  Mystère = 4
  For i = 1 to (X * X + 1)
    If Mystère > 0 Then
      Mystère = - Mystère * X
    Else
      Mystère = X * X + 1
    End If
  Next i
End Function
```

Une seule des expressions suivantes adresse sans ambigüité et sans erreur une cellule (ou une plage de cellules) du tableur. Laquelle ?

A. `Cells(1, 4)`
B. `Sheets("MaFeuille").Range(Cells(1, 2), Cells(3, 4))`
C. `Sheets("MaFeuille").Cells(0, 4)`
D. `Sheets("MaFeuille").Cells("GBA3")`
E. `Sheets("MaFeuille").Range(Sheets("MaFeuille").Cellls(1, 2), Sheets("MaFeuille").Cells(3, 4))`

Une procédure sans argument peut être appelée directement depuis l'interface du logiciel de tableur, typiquement en appuyant sur un bouton ⏯. Ce n'est pas le cas d'une procédure avec argument. Comment exécuter une telle procédure ? Quel est l'intérêt d'une telle procédure ?

```
Dim X, Z As Integer
X = 4
Z = X + 1
X = X + Z
Z = X
X = X + X
Z = Z - 3
X = X - Z
```

Quelle valeur contient `X` après exécution de ces intructions ?

J'écris une fonction qui compte, sur une colonne d'un tableur représentant des noms de famille, le nombre de ces noms qui sont composés.

A priori, quels sont les types les plus adéquats à donner à la variable qui compte ces noms ?

A. `Variant`
B. `Double`
C. `Int`
D. `String`
E. `Long`

Plus difficile ⚠ On considère la fonction suivante :

```
Function Grabadob(X As Double) As Integer
  Grabadob = 0
  While (Grabadob * Grabadob < X)
    Grabadob = Grabadob + 1
  Wend
End Function
```

Quelle est la valeur retournée par `=Grabadob(43)` ? Pour répondre, essayer de déduire le comportement général de la fonction pour toute valeur de `X`.

Si le contenu d'une boucle `For` termine, alors la boucle termine également. Ici, le corps de la boucle ne contient que des tests et des affectations, ainsi la boucle termine.

Il y a une exception concernant les boucles `For` : la variable de comptage introduite par la boucle ne *doit pas* être modifiée à l'intérieur de la boucle (ici, pas d'instruction de la forme `i = ...`), sinon, il est plus difficile de prévoir son comportement.

Ne faites pas ça. ☒

Réponse : Non, elle termine toujours

Une procédure interagit avec l'utilisateur. À l'inverse d'une fonction, elle ne doit pas renvoyer de valeur. De la même manière, une fonction, pour calculer, peut utiliser des variables.

Les variables constituent un espace de travail *local* pour les fonctions comme pour les procédures.

Réponse : A, D

Les deux premières expressions sont ambigües car les appels à `Cells` ne sont pas préfixés d'un `Sheets(...).`. La troisième expression adresse la ligne 0, qui n'existe jamais. Enfin, la dernière expression est mal écrite : `Cellls` au lieu de `Cells`. ☒

La bonne réponse adresse la cellule située à la colonne `"GBA"`, lointaine (c'est la 4785^{e}) mais valide, et à la 3^{e} ligne de la feuille `"MaFeuille"`.

Réponse : D

Lorsqu'on appelle `=Mystère(-3)`, l'expression `X * X + 1` s'évalue à 10. Puis les instructions se répètent pour `i` allant de 1 à 10, soit 10 répétitions.

À noter que puisque $X^2 + 1 \geq 1$ pour tout X, il n'y a pas de problème. Cependant si la valeur de fin de la boucle est plus petite que la valeur initiale, la boucle ne s'exécute pas.

Réponse : 10

Une affectation opère en deux étapes : d'abord on évalue l'expression à droite du signe égal, *puis* on affecte le résultat à la variable à gauche.

En particulier `X = Z` n'est pas la même chose que `Z = X` ! De plus le symbole à gauche doit être un emplacement mémoire : l'instruction `4 = X` n'a pas de sens. Plusieurs langages évitent la confusion en adoptant une notation dissymétrique pour l'affectation : `X <- Z`, `X := Z`, etc.

Réponse : 12

Une procédure avec argument peut être appelé *depuis une autre procédure*, avec le mot-clé `Call` :

```
Call MaProcedure(argument1, argument2, ...)
```

L'intérêt est ici le même que pour des fonctions, en anglais, on parle de *separation of concerns* (« séparation des préoccupations »). Par exemple, une procédure `MiseEnPageCellule(Ligne as Integer, Col as Integer)` qui reçoit en argument un numéro de ligne et de colonne peut s'occuper de modifier la mise en page de la cellule associée (mettre le texte en gras, ajouter une bordure, etc. avec la syntaxe fastidieuse qu'on connait). Puis cette procédure peut être réutilisée par d'autres pour demander directement la mise en page d'une cellule.

Réponse : Avec un mot-clé dédié ; même intérêt que pour une fonction

Le compteur `Grabadob` est incrémenté de 1 à chaque tout de boucle. Il parcourt, jusqu'à la sortie de la boucle, tous les entiers de 1 en 1, depuis 0. La boucle s'arrête lorsque la condition *n'est plus vérifiée*, c'est-à-dire lorsque le carré de `Grabadob` dépasse `X`. Ainsi, pour $X \geq 0$, la fonction retourne le plus petit entier dont le carré est supérieur ou égal à X. (On note ce nombre $\lceil \sqrt{X} \rceil$.)

Sachant que $6^2 = 36 < 43 \leq 49 = 7^2$, on en déduit la valeur de retour.

Réponse : 7

Le type `Variant` (aussi appelés `Any` dans d'autres langages) correspond à n'importe quelle donnée possible. Il ne faut l'utiliser que si nécessaire.

On cherche à stocker un nombre (d'occurrences), donc le type `String` des chaines de caractère n'est pas adéquat.

Utiliser le type `Double` des nombres à virgule serait ici maladroit : la manipulations de ces nombres est plus couteuse, source d'erreurs d'approximation et sans avantage ici.

Enfin, sans connaissance *a priori* sur le nombre d'entrée dans le tableau, les deux types de nombres entiers `Int` et `Long` sont possibles.

Réponse : C, E

Soit une fonction f qui prend en argument un entier et qui renvoie un entier. (Par exemple, la fonction qui prend un nombre et renvoie son double.) On suppose cette fonction implémentée dans VBA avec une fonction qu'on appelle `FonctionF` :

```
Function FonctionF(X As Integer) As Integer
```

Cette fonction reçoit donc en argument un entier `X` et retourne également un entier. On définit la suite $(u_n)_{n\geq 0}$ par la relation suivante : $u_0 = 3; \quad u_{n+1} = f(u_n)$. Ainsi, le premier terme de la suite vaut 3, le deuxième $f(3)$, le troisième $f(f(3))$, et ainsi de suite. En gardant l'exemple d'une fonction qui prend un nombre et renvoie son double, on aurait $u_0 = 3, u_1 = 6, u_2 = 12, u_3 = 24, \ldots$

Écrire en VBA une fonction qui prend en argument un entier n et retourne l'entier u_n.

L'idée centrale est la suivante : il faut répéter une application de fonction. Il faut donc une boucle. On utilisera une variable `U` qui contiendra, à chaque étape, un terme de la suite. La solution n'est pas unique, on propose la fonction suivante :

```
Function FonctionRepetee(N As Integer) As Integer
  Dim U As Integer
  U = 3
  For i = 1 to N
    U = FonctionF(U)
  Next i
  FonctionRepetee = U
End Function
```

On utilise une boucle `For` car on connait « à l'avance » le nombre d'itérations nécessaires. En cas de doute sur les valeurs extrême (`1` ou `0` ? `N` ou `N - 1` ?) :

- testez avec des valeurs concrètes (« que doit renvoyer la fonction si `N` vaut 0 ? ») ;
- rappelez-vous que la boucle `For i = A to B` s'évalue 1 fois quand `A` et `B` sont égaux, et 0 fois quand `B` est plus petit que `A`.

Réponse : Voir la fonction ci-après