Programmation Impérative
1ère Année – Session 1

6 janvier 2022.
Durée 1h30.

Feuille de notes A4 recto-verso autorisée,
Sans autre document.

— Tous les programmes demandés **seront écrits soit en langage algorithmique, soit en ADA.**
— Ces programmes devront respecter scrupuleusement **TOUTES les consignes de bonne programmation définies en cours, TD et TP.**

1 Questions de cours. (5 points)

C1. Voici le raffinage R1 du problème d'affichage des diviseurs entiers de N.

```
R1 : Comment "Afficher les diviseurs entiers de N, N lu au clavier avec N>=1"

     Lire l'entier N avec N>=1                    N : [F1] Entier
     Déterminer et afficher les diviseurs entiers N : [F2]
```

Le flot de la donnée N est représenté sur les deux actions.
Indiquer le 'sens' du flot qui doit remplacer les balises `[F1]` et `[F2]`.

C2. Quelles sont les structures de contrôle de type répétition dans notre langage algorithmique ? Quels arguments donner pour aider à choisir la bonne répétition ?

C3. Identifier les différences entre la spécification et l'implantation d'un module en argumentant la réponse en quelques lignes.

C4. Qu'est ce qu'un module générique ? Quel est l'intérêt ? Exemples ?

C5. Soient le type `P_Entier EST POINTEUR SUR Entier` et les variables `P1, P2 : P_Entier`.
Soient les instructions suivantes :

```
P1 <-- Null
P2 <-- NEW Entier
P2^ <-- 21
P1 <-- P2
```

(a) Indiquer le type et la valeur de la donnée enregistrée dans la variable P1.

(b) Schématiser l'état de la mémoire et la valeur des variables à la fin de l'exécution de ces instructions.

2 Modélisation. (1.5 points)

On souhaite modéliser un jeu de dés qui utilise 3 dés à six faces. Chaque dé a une couleur différente. On supposera par exemple qu'on a un dé rouge, un dé vert et un dé jaune. Pendant le jeu, chaque joueur lance à tour de rôle les 3 dés en même temps.

Q1. Proposer la définition d'un type `T_DES_3_COULEUR` qui permette d'enregistrer la valeur des 3 dés à chaque lancé dans une variable unique.
Vous pourrez définir et composer différents types pour proposer une modélisation lisible et cohérente.

3 Sous-programmes. (2.5 points)

Considérons les entêtes (signatures) des deux sous-programmes suivants.

```
FONCTION F(X : IN ENTIER) RETOURNE ENTIER

PROCEDURE G(X : IN OUT ENTIER ; N : IN ENTIER)
```

Soient `A`, `M`, `P` et `Y` des variables de type `ENTIER`, déjà initialisées, et considérons les appels à ces sous-programmes

```
1. F (2 * Y)

2. G (2 * M, F (P))

3. G (P, A * P)

4. F (F (M))

5. F (G (Y, A))
```

Q1. Pour chacun des appels précédents, indiquer si l'appel est correct ou non. Les réponses doivent être justifiées.

4 Sous-programmes et exceptions. (5 points)

Cet exercice s'intéresse au calcul des racines **réelles** d'un polynôme du second degré.

Q1. Spécifier un sous-programme qui retourne les deux racines **réelles** $r1 = \frac{-b+\sqrt{\Delta}}{2a}$ et $r2 = \frac{-b-\sqrt{\Delta}}{2a}$ pour le polynôme $ax^2 + bx + c$, avec $\Delta = b^2 - 4ac$ et a, b et c des réels.
Le sous-programme lèvera des exceptions si le polynôme n'est pas de degré 2 ($a = 0$) ou si les racines ne sont pas réelles ($\Delta < 0$).

Q2. Spécifier et définir un sous-programme `Principal` qui invite un utilisateur à renseigner la valeur des coefficients a, b et c réels du polynôme et qui affiche le résultat du calcul des deux racines à l'écran.
Ce sous-programme utilise celui de la question Q1 pour déterminer les racines.
Il est attendu de votre implantation qu'elle utilise le mécanisme des exceptions :

(a) pour signaler à l'utilisateur toute erreur de type sur les données entrées.

(b) pour signaler qu'il s'agit d'une équation du premier degré et non du second.

(c) pour signaler qu'il n'y pas de racine réelle.

Dans les deux premiers cas, l'utilisateur devra saisir à nouveau les 3 coefficients.

5 Allocation dynamique et liste chaînée. (6 points)

Soit la structure de données simplement chaînée suivante :

```
TYPE T_Liste EST POINTEUR SUR T_Cellule

TYPE T_Cellule EST ENREGISTREMENT
   Élément : Entier   -- Élément rangé dans la cellule
   Suivante : T_Liste -- Accès à la cellule suivante
FIN ENREGISTREMENT
```

Cette liste linéaire simple peut contenir plusieurs occurrences d'un même élément. Pour toutes les questions de cet exercice, vous utiliserez une approche de programmation offensive.

Q1. Spécifier et définir un sous-programme d'initialisation d'une telle liste.

Q2. Spécifier et définir un sous-programme d'ajout d'un élément dans la liste. Ce nouvel élément est inséré en tête de liste.

Q3. Spécifier et définir une version **itérative** d'un sous-programme qui compte le nombre d'occurrences d'un même élément donné en paramètre.

Q4. Spécifier et définir une version **récursive** d'un sous-programme qui détruit toutes occurrences d'un même élément donné en paramètre.

Q5. L'élément rangé dans une cellule est ici un entier. Quels changements doit-on réaliser pour rendre le type de l'élément générique ?