PIM : Mini-projet 1

TODO : Vous ne pouvez pas modifier ce document. Vous devez en **créer une copie** (Fichier / Créer une copie) que vous **partagerez avec votre enseignant de TP** (c'est lui qui corrigera votre mini-projet).
TODO : Nommer votre document PIM-MP1-X-Nom où X est la lettre de votre groupe de TP (A, B, C…) et Nom votre nom.
TODO : Les TODO doivent être enlevés quand ils sont traités.

Raffinages

TODO : écrire ici les raffinages en suivant les règles présentées en cours. On ne donnera pas d'exemples.

R0 : Jouer au jeu d'allumettes .

R1 : Comment "Jouer au jeu d'allumettes ."?

Demander le niveau du jeu du l'ordinateur . **niveau :in out Caractère**
Demander au joueur s'il veut commencer .
Nombre_d'allumettes <-13 **Nombre_d'allumettes : in out entier**
Répéter
    Si Tour_Humain alors **Tour_Humain:in out Booleen**
        Jouer pour l'humain
    Sinon si non Tour_Humain alors
        Jouer pour l'ordinateur
Finsi
Si Action_Valide alors **Action_Valide : in out Booleen**
    Mettre à jour le Nombre d'allumettes
Fi
Jusqu'à Nombre d'allumettes =0
FinRépéter

Afficher le gagnant

R2 : Comment “Demander le niveau de jeu du de l’ordinateur “ ?

Ecrire (“ Niveau de l'ordinateur (n)aïf, (d)istrait, (r)apide ou (e)xpert ?“)
Lire(Niveau) Niveau : in out

R2 : Comment “traiter le niveau du jeu du l’ordinateur” ?

```
Selon Niveau Faire
        ‘n', 'N'  =>  choisir le niveau naif
        ‘d', 'D'  => choisir le niveau distrait
        ‘r', 'R'   => choisir le niveau rapide
        ‘e', 'E'   => choisir le niveau expert
        Autres  => choisir le niveau expert
FinSelon
```

R2 : Comment “Demander au joueur s’il veut commencer” ?

Ecrire ("Est-ce que vous commencez (o/n)")
Lire(choix) choix : out

R2 : Comment “traiter le choix du joueur “ ?

```
Si choix=o alors
                Tour <- True .
Sinon si choix=O alors
                Tour <- True.
Sinon si choix=n alors
                Tour<- False.
Sinon si choix=N alors
                Tour<- False.
FinSI
```

R2 : Comment “Mettres à jour le Nombre d’allumettes.”?

```
Si Tour_humain :
        Nombre_d’allumettes <- Nombre_d’allumettes - allumettes
FinSi
        Allu_1<-Nombre_d’allumettes              allu_1: in out entier
        Pour i de 1 JusquA 3 Faire
                TantQue allu_1  > = 5
                        Ecrire en meme ligne (‘| | | | |  ‘)
                        Nombre d’allu_1 <- Nombre d’allumettes  -5
                FinTantQue
                Si allu_1 >0 :
                        Pour i de 0 jusquA allu_1 fois  Faire
                                Ecrire en meme ligne (‘| ‘)
```

```
                    FinPour
                FinSi
            FinPour
```

R2 : Comment “Afficher le gagnant ” ?

```
Si Tour_Humain Alors :
    Ecrire(“Vous avez gagné”)
Sinon si Alors  :
    Ecrire(“J’ai gagné”)
FinSi
```

R2 : Comment “Jouer pour l’humain ” ?

```
Action_Valide<-True
Afficher les 13 allumettes en groupes de 5 .
Demander à l’humain combien d’allumettes il veut prendre .
Tour_Humain<-False
```

R2 : Comment “Jouer pour l’ordinateur .“ ?

```
Action_Valide<-True
SI niveau=naif alors
    Choisir aléatoirement entre 1 et 3 allumettes
Sinon si niveau=distrait alors
    Choisir aléatoirement entre 1 et 3 sans tenir compte du nombre restant
Sinon si niveau=rapide alors
    Prendre systématiquement le maximum possible
Sinon si niveau=expert alors
    Calculer le meilleur coup pour laisser l'adversaire en désavantage.
FinSi
Ecrire(“Je prends “Nombre_d’allumettes” allumettes”)
Tour_Humain <-True
```

R3 : Comment “Choisir aléatoirement entre 1 et 3 allumettes “ ?

```
Si Nombre_d’allumettes=1 alors
    Nombre_d’allumettes= Nombre_d’allumettes - 1
Sinon si Nombre_d’allumettes =2 alors
    Nombre_d’allumettes = Nombre_d’allumettes -aléa(1,2)
Sinon si Nombre d’allumettes >=3 alors
    Nombre_d’allumettes = Nombre_d’allumettes -aléa(1,3)
FinSi
```

R3 : Comment “Choisir aléatoirement entre 1 et 3 sans tenir compte du nombre restant “ ?

```
Nombre_aleatoire <- aléa(1,3) Nombre_aleatoire : in out entier
TantQue Nombre_d'allumettes <Nombre_aleatoire  Faire
        Ecrire("Arbitre : Il reste seulement "Nombre d'allumettes " allumettes.")
        Nombre d'allumettes <- Nombres d'allumettes -aléa(1,Nombre d'allumettes)
FinTantQue
Nombre d'allumettes <- Nombre d'allumettes -Nombre_aleatoire
Ecrire("Je prends "Nombre d'allumettes" allumettes")
```

R3 : Comment "Prendre systématiquement le maximum possible" ?

```
Si Nombre d'allumettes >3 Alors
        Nombre d'allumettes <- Nombre d'allumettes - 3
Si Nombre d'allumettes =2 :
        Nombre d'allumttes <- Nombre d'allumettes - 2
SI Nombre d'allumettes = 1 :
        Nombre d'allumettes <- Nombre d'allumettes - 1
FinSi
```

R3 : Comment "Calculer le meilleur coup pour laisser l'adversaire en désavantage." ?

```
        R<-Nombre d'allumettes%4            R:in entier
        Si R=0 alors
                Nombre d'allumettes<- Nombre d'allumettes - 3
        Sinon si R=1 alors
                Nombre d'allumettes <-Nombre d'allumettes-1
        Sinon si R=2 alors
                Nombre d'allumettes <- Nombre d'allumettes -1
        Sinon si R=3 alors
                Nombre d'allumettes <- Nombre d'allumettes - 2
        FinSi
```

R3 : Comment "Demander combien d'allumettes l'humain veut prendre ." ?

```
Ecrire ("Combien d'allumettes prenez-vous ?")
Lire (allumettes ) allumettes: in out entier
Si allumettes >3 alors
        Ecrire ("Arbitre :Il est interdit de prendre plus de 3 allumettes.")
        Action<-False
Sinon si nombre d'allumettes <allumettes alors
        Ecrire (" Arbitre :Il reste seulement nombre allumettes allumettes.")
        Action_Valide<-False
Sinon si nombre d'allumettes=0 alors
        Ecrire ("Arbitre : Il faut prendre au moins une allumette.")
        Action<-False
```

Sinon si Nombre d’allumettes >allumettes Alors
Nombre d’allumettes <- Nombre d’allumettes -allumettes

Évaluation des raffinages par l'étudiant

		Evaluation Etudiant (I/P/A/+)	Justification / commentaire	Evaluation Enseignant (I/P/A)
Forme (D-21)	Respect de la syntaxe Ri : Comment "... une action complexe ..." ? des actions combinées avec des structures de controle Rj : ...	P		
	Verbe à l'infinitif pour les actions complexes	A	Tous les verbes sont à l'infinitif	
	Nom ou équivalent pour expressions complexes	P	Je ne sais pas si c'est parfait	
	Tous les Ri sont écrits contre la marge et espacés	A	oui	
	Les flots de données sont définis	P	Je ne sais pas si c'est parfait	
	Une seule structure de contrôle par raffinage	P	Je ne sais pas si c'est parfait	
	Pas trop d'actions dans un raffinage (moins de 6)	A	Oui	
	Bonne présentation des structures de contrôle	P	Je ne sais pas si c'est parfait	
Fond (D21-D22)	Le vocabulaire est précis	P	Je ne sais pas si c'est parfait	
	Le raffinage d'une action décrit complètement cette action	P	Je ne sais pas si c'est parfait	
	Le raffinage d'une action ne décrit que cette action	P	Je ne sais pas si c'est parfait	
	Les flots de données sont cohérents	I	Je ne sais pas si c'est parfait	
	Pas de structure de contrôle déguisée	P	Je ne sais	

			pas si c'est parfait	
	Qualité des actions complexes	P	Je ne sais pas si c'est parfait	

Remarques diverses

TODO : Indiquer ici ce qui est utile à l'enseignant pour comprendre les raffinages. Cette partie peut être vide.

Évaluation du code

		Consigne : Mettre O (oui) ou N (non) dans la colonne Etudiant suivant que la règle a été respectée ou non. Une justification peut être ajoutée dans la colonne "commentaire".	
Commentaire	**Etudiant (O/N)**	**Règle**	**Enseignant (O/N)**
	o	Le programme ne doit pas contenir d'erreurs de compilation.	
	o	Le programme doit compiler sans messages d'avertissement.	
	o	Le code doit être bien indenté.	
	o	Les règles de programmation du cours doivent être respectées : toujours un Sinon pour un Si, pas de sortie au milieu d'une répétition...	
	o	Pas de code redondant.	

	o	On doit utiliser les structures de contrôle adaptées (Si/Selon/TantQue/Répéter/Pour)	
	o	Utiliser des constantes nommées plutôt que des constantes littérales.	
	o	Les raffinages doivent être respectés dans le programme.	
	o	Les actions complexes doivent apparaître sous forme de commentaires placés AVANT les instructions correspondantes, avec la même indentation	
	o	Une ligne blanche doit séparer les principales actions complexes	
	o	Le rôle des variables doit être explicité à leur déclaration (commentaire).	