Devoir d'informatique

D'après Anthony MDF.

1 Introduction

2 Prise en main

2.1 Résolution d'un problème

Question 1 *Résoudre le problème, pas à pas, comme précédemment avec les entrées* `argent = 15`, `prix = 1`, `emballage = 3`.

Correction Nous achetons et mangeons 15 chocolats. Nous retournons 15 emballages et recevons 5 autres chocolats. On retourne 3 emballages, on prend 1 chocolat et on le mange. (en gardant 2 emballages). Maintenant, nous avons 3 emballages. Retournez 3 emballages et obtenez 1 chocolat de plus. Donc le total des chocolats vaut $15+5+1+1=22$.

Question 2 *Justifier que le type* `int` *est un type adapté au traitement de ce problème (par opposition au type* `float`*).*

Correction Le type entier est un bon type puisqu'il présente une large gamme de valeurs accessibles —2 147 483 648 à 2 147 483 647 et que les calculs effectués avec ce type de variable sont toujours très rapides. De plus il n'y aura pas d'erreur de précision dues aux approximations de la représentation des nombres flottants. Avouons que le nombre d'itérations est tellement faible que ce problème est un faux problème ici ; cette question sert globalement de question de cours ici et n'est pas totalement justifiable par une bonne raison.

2.2 Étude d'une suite

Soit la suite $(u_n)_{n\in\mathcal{N},n\geq 0}$ définie par $u_{n+1} = u_n//p + u_n\%p$ avec u_0 et p deux entiers positifs non nuls.

Question 3 *Démontrer que cette suite est une suite d'entiers naturels strictement décroissante pour $p > 1$. Que se passe-t-il dans le cas où $p = 1$?*

Correction On remarque deux choses :

- $u_{n+1} = u_n//p + u_n\%p = q_n + r_n$;
- $u_n = pq_n + r_n$

Alors en partant de $p > 1$, on obtient :

$$\begin{aligned} q_n p &> q_n \\ q_n p + r_n &> q_n + r_n \\ u_n &> u_{n+1} \end{aligned}$$

Dans le cas où $p = 1$, la suite est constante (et pour la suite, dans ces conditions, cette suite n'atteindra jamais son cas de base).

3 Résolution du problème

3.1 Résolution par fonction récursive

Question 4 *Proposer une fonction récursive* `def chocolatSuppRecursif(nbChocolat:int, emballage:int) -> chocolatSupplementaire:int :` *prenant en argument un nombre de chocolats et le nombre d'emballages nécessaires pour obtenir un chocolat supplémentaire et renvoyant le nombre de chocolats supplémentaires obtenus.*

Correction

```
def chocolatSuppRecursif(nbChocolat, emballage) :
    # Cas de base
    if (nbChocolat < emballage) :
        return(0)
    # Recursion
    nouveauChocolat = nbChocolat // emballage
    resteEmballage = nbChocolat % emballage
    return(nouveauChocolat + chocolatSuppRecursif(nouveauChocolat + resteEmballage, ↙
        emballage))
```

On retrouve « évidemment » la suite proposée au début de ce problème.

Question 5 *Proposer une fonction* `def combienTotalChocolat(argent:int, prix:int, emballage:int) -> nbTotalChocolat:int :` *prenant en argument les trois paramètres définis dans la sous-section* **??** *et renvoyant le nombre total de chocolats obtenus.*

Correction

```
def combienTotalChocolat(argent, prix, emballage) :
    nbChocolatPremierTour = argent // prix
    return(nbChocolatPremierTour + chocolatSuppRecursif(nbChocolatPremierTour, emballage))
```

Question 6 *Combien d'emballages restera-t-il ? Écrire la (les) ligne(s) de script permettant d'évaluer* `nbEmballage` *pour les trois paramètres définis précédemment.*

Correction `nbEmballage=combienTotalChocolat(argent:int, prix:int, emballage:int)%emballage`

Dans le cas où `emballage = 1`, on obtient le message d'erreur suivant `RecursionError: maximum recursion depth exceeded in comparison`.

Question 7 *Pourquoi obtient-on ce message ? Comment évolue le code si on change le* `maximum recursion depth` *?*

3.2 Version itérative

On souhaite maintenant résoudre le problème de manière itérative.

Question 8 *Proposer une fonction itérative* `def combienIteratif(argent:int, prix:int, emballage:int) -> nbTotalChocolat:int :` *prenant en argument les trois paramètres définis dans la sous-section* **??** *et renvoyant le nombre total de chocolats obtenus.*

Correction La logique reste la même ; il suffit d'introduire un `while`.

```
def combienIteratif(argent, prix, emballage) :
    nbChocolatPremierTour = argent // prix
    nbEmballage = nbChocolatPremierTour
    nbTotalChocolatSupp = 0
    while nbEmballage >= emballage :
        nvChoco = nbEmballage // emballage
```

```
        nbEmballage = nvChoco + nbEmballage%emballage
        nbTotalChocolatSupp += nvChoco
    return(nbChocolatPremierTour + nbTotalChocolatSupp)
```

4 Représentation des nombres

Soit un sytème d'exploitation ou les nombres sont codés sur 8 bits.

Question 9 *Combien d'entiers positifs sont codables ? Quel est le plus petit entier codable ? Quel est le plus grand ?*

Correction On peut coder 256 nombres entre 0 et 255.

Question 10 *On souhaite coder des entier positifs et négatifs. Quel est le plus petit entier codable ? Quel est le plus grand ?*

Correction On peut coder 256 nombres entre -128 et 127.

Question 11 *Coder le nombre 23 en binaire. Coder le nombre -23 en binaire.*

Correction 10111

Question 12 *Coder 101100 en décimal.*

Correction 44

Question 13 *Coder 101100 en hexadécimal.*

Correction 2c

Question 14 *Coder A9 en décimal.*

Correction 169