TD

Sources :
exercice 3 : Recherche d'invariant exercice 4 : Recherche d'invariant exercice 5 : Recherche d'invariant

Savoirs et compétences :

- *AN.C1 : Justifier qu'une itération (ou boucle) produit l'effet attendu au moyen d'un invariant*
- *AN.C2 : Démontrer qu'une boucle se termine effectivement*
- *AN.S1 : Recherche dans une liste, recherche du maximum dans une liste de nombres, calcul de la moyenne et de la variance.*
- *AN.S2 : Recherche par dichotomie.*
- *AN.S4 : Recherche d'un mot dans une chaîne de caractères.*

Proposition de corrigé

0.1 Test de primalité

Q 1 : Proposer un invariant de boucle pour démontrer cet algorithme.

```python
def est_premier(n):
    """ Renvoie True si n est premier, False sinon
        Préconditon : n est un entier."""
    for d in range(2,n):
        # n n'est pas divisible par 2, 3, ..., d-1.
        if n % d == 0:
            return False
    return True
```

0.2 Fonction mystère

Q 1 : À chaque tour de boucle, k est incrémenté de 1 et p est multiplié par b.

Au début du premier tour de boucle, on a $k=0$ et $p=1$. Au tour suivant, $k=1$ et $p=b$. Au second tour, $k=2$ et $p=b^2$. Le tableau se dresse aisément.

Q 2 : Montrons que « $b=p^k$ » est un invariant pour la boucle `while`. En entrée de boucle, on a $k=0$ et $p=1=b^0$, donc l'invariant est bien initialisé. Supposons qu'au début d'un tour de boucle on ait $b=p^k$. À la fin de la ligne 5, comme k est incrémenté de 1, $p=b^{k-1}$. À la fin de la ligne 6, comme p est multiplié par b, on a $p=b^k$. L'invariant est donc vrai au début du tour de boucle suivant.

Ainsi, « $b=p^k$ » est un invariant pour la boucle `while`.

Q 3 : Montrons que « $a-p$ » est un variant pour la boucle `while`.

- C'est bien un entier car par construction p est toujours un entier et a est entier par définition.
- C'est un entier positif : d'après la condition de la boucle `while`, p divise a. Comme a est strictement positif et comme p est positif, $p \leq a$, donc $a-p \geq 0$.
- C'est un entier positif qui décroît strictement à chaque tour de boucle : à chaque tour de boucle, p est multiplié par b et $b>1$.

Q 4 : On a écrit un variant pour la boucle `while`, donc la fonction renvoie bien un résultat. Par l'invariant, en sortie de boucle, on a $p=b^k$. En sortie de boucle, k est le plus petit entier pour lequel b^k ne divise pas a. On renvoie en sortie $k-1$. Ainsi, la fonction renvoie le plus grand entier k tel que b^k divise a.

0.3 Invariant pour les calculs de moyennes et de variances

Q 1 : Soit les algorithmes de calculs de moyenne ci-dessous, proposez des invariants de boucles.

```python
def moyenne(t):
    """Calcule la moyenne de t
       Précondition : t est un tableau de
       nombres non vide"""
    s = 0
    for x in t:
        # Invariant :
        # s == somme des éléments de t
        # avant x
        s = s + x
    return s/len(t)
```

```python
def moyenne(t):
    """Calcule la moyenne de t
       Précondition : t est un tableau de
       nombres non vide"""
    n = len(t) # Longueur de t
    s = 0
    for i in range(n):
        # Invariant : s == sum(t[0:i])
        s = s + t[i]
    return s/n
```

Q 2 : Soit l'algorithme de calculs de variance ci-dessous, proposez un invariants de boucles.

```python
def variance(t):
    """Renvoie la variance de t
       Précondition : t est un tableau de
       nombres non vide"""
    sc = 0
    for x in t:
        # Invariant : sc == somme des
        # carrés des éléments de
        # t avant x
        sc = sc + x**2
    return sc/len(t) - moyenne(t)**2
```

0.4 Invariant pour les recherches de maximum de tableaux

Q 1 : Soit les algorithmes de recherche de maximum d'un tableau ci-dessous, proposez des invariants de boucles.

```python
def maxi(t):
    """Renvoie le plus grand élément de t.
       Précondition : t est un tableau
       non vide"""
    m = t[0]
    for x in t:
        # Invariant : m est le plus grand
        # élément trouvé jusqu'ici
        if x > m:
            m = x # On a trouvé plus grand,
                  # on met àjour m
    return m
```

```python
def maxi(t):
    """Renvoie le plus grand élément de t.
       Précondition : t est un tableau
       non vide"""
    m = t[0] # Initialisation par le
             # premier élément
```

```python
    for i in range(1, len(t)):
        # Invariant : m == max(t[0:i])
        if t[i] > m:
            m = t[i] # On a trouvé plus
                     # grand, on met à
                     # jour m
    return m
```

Q 2 : Soit les algorithmes de recherche d'indice de maximum d'un tableau ci-dessous, proposez des invariants de boucles.

```python
def indicemaxi(t):
    """Renvoie l'indice du plus grand
       élément de t.
      Précondition : t est un tableau
      non vide"""
    im = 0 # Indice du maximum,
          # initialisation par
          # le premier élément
    for i in range(1, len(t)):
        # Invariant : im est indice d'un
        # plus grand élément de t[0:i]
        if t[i] > t[im]:
            im = i # On a trouvé plus grand,
                   # on met àjour im
    return im

def indicemaxi(t):
    """Renvoie l'indice du plus grand élément
      de t.
      Précondition : t est un tableau
      non vide"""
    im = 0 # Indice du maximum,
           # initialisation par le
           # premier élément
    for i, x in enumerate(t):
        # Invariant : im est indice
        # d'un plus grand élément de t[0:i]
        if x > t[im]:
            im = i # On a trouvé plus grand,
                    # on met àjour im
    return im
```

0.5 Invariant pour les recherches d'occurrences dans un tableau

Q 1 : Soit les algorithmes de test d'appartenance d'un élément dans un tableau. Proposer un invariant de boucle.

```python
def appartient(e, t):
    """Renvoie un booléen disant si e
    appartient àt
      Précondition : t est un tableau"""
    for x in t:
        # Invariant : e n'est pas positionné
        # dans t avant x
        if e == x:
            return True # On a trouvé e,
                        # on s'arrête
    return False
```

Q 2 : Soit les algorithmes de recherche d'indice de première occurrence d'un élément dans un tableau. Proposer un invariant de boucle.

```python
def ind_appartient(e,t):
    """Renvoie l'indice de la première
       occurrence de e dans t,
       None si e n'est pas dans t
       Précondition : t est un tableau"""
    for i in len(t):
        # Invariant : e n'est pas dans t[0:i]
        if t[i] == e:
            return i # On a trouvé e
                     # à l'indice i
    return None
```