Algorithmique
Correction Contrôle n° 3 (C3)

INFO-SPÉ (S3#) – EPITA

6 avril - 10h

Solution 1 **(Quelques questions –** ***4 points*****)**

1. Les méthodes de hachage de base sont : extraction, compression, division, multiplication.
2. Le hachage linéaire ou le double hachage.
3. Le hachage avec chaînage séparé. En effet les éléments sont chaînés entre eux à l'extérieur du tableau.
4. La recherche par intervalle est incompatible avec la dispersion des éléments effectuée par le hachage.

Solution 2 **(Sérialisation –** ***4 points*****)**

1. *Le vecteur de pères :*

1	2	3	4	5	6	7	8	9	10	11	12
3	3	10	3	3	10	8	9	10	−1	7	7

2. *La fonction* `buildParentVect(T, n)` *:*

```
        def __buildParentVect(T, P):
            if T != None:
                C = T.child
                while C != None:
                    P[C.key] = T.key
                    __buildParentVect(C, P)
                    C = C.sibling

        def buildParentVect(T, n):
            P = [-1] * n
            __buildParentVect(T, P)
            return P

```

Solution 3 **(Représentation par listes –** ***5 points*****)**

```
        def __fromList(s, i=0):
            i += 1
            key = ""
            while s[i] not in "()":
                key += s[i]
                i += 1
            T = Tree(int(key))
            T.children = []
            while s[i] != ')':
                (Child, i) = __fromList(s, i)
                T.children.append(Child)
            i = i + 1
            return (T, i)

        def fromList(L):
            (T, _) = __fromList(L)
            return T
```

Solution 4 **(B-arbre : suppression –** ***2 points*****)**

1. L'arbre de l'énoncé est un B-arbre de degré minimal (ordre) 3.

2. Après suppresion de la valeur 72, avec le principe "à la descente" :

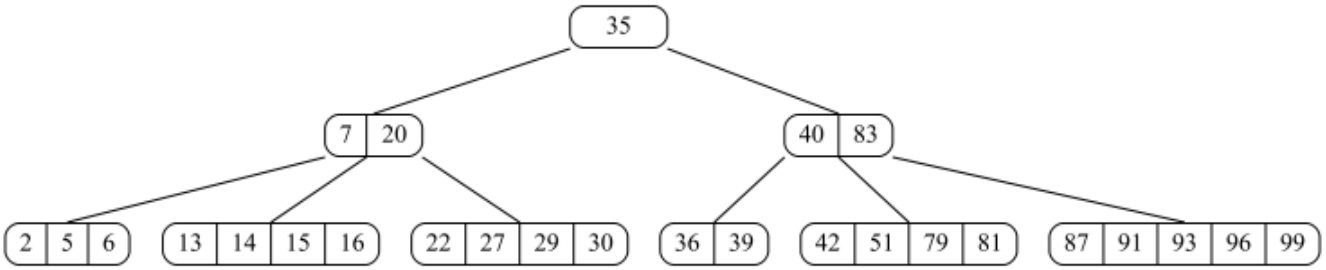

FIGURE 1 – Après suppression

Solution 5 **(Au suivant –** ***5 points*****)**

```
        def findNextNath(B, x):
            vNext, i = None, 0
            while i < B.nbKeys and B.keys[i] <= x:
                i += 1
            if B.children == []:
                if i < B.nbKeys:
                    return B.keys[i]
                else:
                    return None
            else:
                vNext = findNextNath(B.children[i], x)
                if vNext != None:
                    return vNext
                elif i < B.nbKeys:
                    return B.keys[i]
                else:
                    return None

        # Student's
        def findNextStu(B, x, y = None):
            i = 0
            while i < B.nbKeys and B.keys[i] <= x:
                i += 1
            if i < B.nbKeys:
                y = B.keys[i]
            if B.children == []:
                return y
            else:
                return findNextStu(B.children[i], x, y)
```