Algorithmics
Correction Final Exam #1 (P1)

Undergraduate 1^{st} year S1# – Epita

19 juin 2018 - 9 : 00

Solution 1 (Searching algorithms – *3 points*)

1. Linear search regardless of element order: 13
2. Linear search taking into account the element order: 9
3. Binary search: $8 = 2 \times 4$

Solution 2 (Séquences et ABR – *3 points*)

1. Séquences valides :

   ☒ 50, 70, 2048, 75, 1500, 1024 **oui**

   □ 50, 75, 2048, 70, 1500, 1024

   ☒ 2048, 50, 70, 75, 1500, 1024 **oui**

   □ 50, 75, 70, 2048, 1500, 1024

2. *Principe :*

   As we progress in the list we test that elements are in he actual interval $[inf, sup]$. If not the sequence is not valid.

   If the actual value is followed by a higher one, $inf \leftarrow x$ (we "go" right), else $sup \leftarrow x$ (we go left).

Solution 3 (Types abstraits - *3 points*)

1. *Quel est le nom de l'opération* mystère *?* **inverse**

2. **Specifications:**

   La fonction `mystery(`L`)` inverse les éléments de la liste L.

```python
def mystery(L):
    n = len(L)
    for k in range(n // 2):
        (L[k], L[n-k-1]) = (L[n-k-1], L[k])
```

Solution 4 (What is it? – *3 points*)

1. (a) `what([(0,0), (10,10), (20,20), (30,30)], 15)` $\boxed{15.0}$

   (b) `what([(0,0), (10,20), (20,40), (30,60)], 24)` $\boxed{12.0}$

   (c) `what([(0,0), (1, 10), (2,100), (3, 1000)], 2.5)` $\boxed{0.25}$

   (d) `what([(0,3), (1,6), (2,9), (3,10), (4,15)], 20)` $\boxed{5.0}$

2. If we consider pairs are coordinates in increasing order, `what(`L`, `Y`)` computes the abscissa X corresponding to the ordinate Y computed by linear interpolation.

Solution 5 (Select Sort (Tri par sélection) – *8 points*)

1. The function `minimum(`L`, `d`, `f`)` returns the position of the minimum value in the list L between the positions d and f, both included (with $0 \leq d < f < len(L)$).

```python
        def minimum(L, d, f):
            pos = d
            for i in range(d + 1, f + 1):
                if L[i] < L[pos]:
                    pos = i
            return pos


        # a nice version
        def minimum2(L, d, f):
            while (d < f):
                if L[d] < L[f]:
                    f = f - 1
                else:
                    d = d + 1
            return d
```

2. The function `selectsort(`L`)` sorts **in place** the list L in increasing order.

```python
def selectSort(L):
    n = len(L)
    for i in range(n - 1):
        pos = minimum(L, i, n - 1)
        (L[i], L[pos]) = (L[pos], L[i]) # swap
```

3. Let L be a list of length n, the select sort performs:

   (a) $\boxed{\frac{n(n-1)}{2}}$ comparisons ;

   (b) $\boxed{2(n-1)}$ element copies.