Algorithmics
Correction Final Exam #1 (P1)

Undergraduate 1^{st} year S1 – Epita

9 Jan. 2018 - 10 : 00

Solution 1 **(Stack or queue? –** ***2 points*****)**

	stack	queue	neither
A B C D E F	√	√	
B D E F A C			√

	stack	queue	neither
D E C B F A	√		
F E D C B A	√		

Solution 2 **(Binary Search –** ***3 points*****)**

1. Decision tree learning of a binary search:

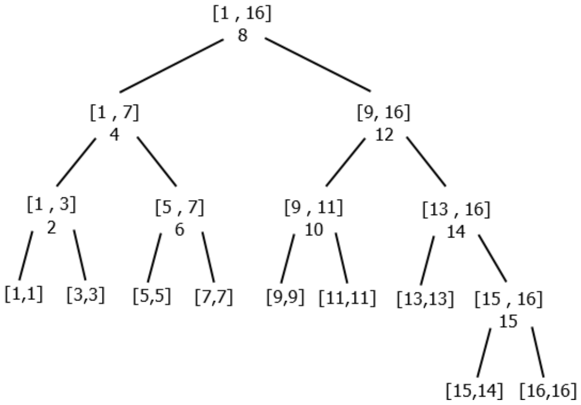

Figure 1: Decision tree learning of a binary search

Each node represents a range of search (left and right bounds) and the rank calculated from the median. Here we use a version of the algorithm that stops when bounds intersect or become equal.

2. (a) Comparison number (integer): $32 = 2 \times (15 + 1)$ (b) List length: 65536 (32768×2)

($log_2(32768) = 15$)

Solution 3 (Algo → Python – *4 points*)

1. **Specifications:**

   The function `test(`L tests whether the list L is sorted by increasing order.

2. The Python function:

```python
def test(L):

    i = 0
    n = len(L)

    while (i < n-1) and (L[i] <= L[i+1]):
        i = i+1

    return (i >= n - 1) # or ==
```

Solution 4 (Minimaxi – *3 points*)

Specifications:

The function `posMiniMaxi(`M`)` returns the pair ($mini$, $maxi$): positions of the minimum and the maximum values of the list L. If the list is empty it raises an exception.

```python

def posMiniMaxi(L):

    if L == []:
        raise Exception("empty list")

    (pMini, pMaxi) = (0, 0)

    for i in range(1, len(L)):
        if L[i] > L[pMaxi]:
            pMaxi = i
        elif L[i] < L[pMini]:
            pMini = i

    return (pMini, pMaxi)
```

Solution 5 **(Merge sort –** ***2,5 + 5 + 2,5 points*****)**

1. **Specifications:**

   The function `partition` splits the list L into two lists of almost identical lengths: one half in each list.

```python
def partition(L):

    n = len(L)
    L1 = []
    for i in range(0, n//2):
        L1.append(L[i])

    L2 = []
    for i in range(n//2, n):
        L2.append(L[i])

    return (L1, L2)
```

2. **Specifications:**

   The function `merge`$(L1,\ L2)$ merges the two sorted in increasing order lists $L1$ and $L2$ into one sorted list.

```python
def merge(L1, L2):

    R = []
    i = j = 0
    n1 = len(L1)
    n2 = len(L2)

    while (i < n1) and (j < n2):
        if L1[i] <= L2[j]:
            R.append(L1[i])
            i = i+1
        else:
            R.append(L2[j])
            j = j+1

    for i in range(i, n1):
        R.append(L1[i])
    for j in range(j, n2):
        R.append(L2[j])

    return R
```

3. **Specifications:**

   The function `mergeSort(`L sorts the list L in increasing order (not "in place": the function builds and returns a new list.)

```python
def mergesort(L):

    if len(L) <= 1:
        return L

    else:
        (L1, L2) = partition(L)

        return merge(mergesort(L1), mergesort(L2))
```