Algorithmics
Correction Final Exam #1 (P1)
(Teacher version)

Undergraduate 1^{st} year S1# – Epita

20 Juin 2019

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 **(Searching algorithms -** ***3 points*****)**

	Sequential search			*Binary search*		
	cost = 1	maximum cost		cost = 1	maximum cost	
	value ?	value ?	cost ?	value ?	value ?	cost ?
(a) $n = 20$	u_0	u_{19}	20	u_9 or u_{10}	u_0	5 or 9
(b) $n = 100$	u_0	u_{99}	100	u_{49} or u_{50}	u_0	7 or 13

***Solution 3* (Sorted – *3 points*)**

The function `is_sorted(L)` checks if the elements of the input list L are sorted in increasing order.

***** 3 pts *****

```
1
2  def is_sorted(L) :
3    i = 0
4    n = len(L) - 1
5    while i < n and L[i] <= L[i+1]:
6      i += 1
7    return i >= n
```

***Solution 4* (Merge sort – *10 points*)**

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

```
1         def partition(L):
2
3             n = len(L)
4             L1 = []
5             for i in range(0, n//2):
6                 L1.append(L[i])
7
8             L2 = []
9             for i in range(n//2, n):
10                L2.append(L[i])
11
12            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.

```
1         def merge(L1, L2):
2
3             R = []
4             i = j = 0
5             n1 = len(L1)
6             n2 = len(L2)
7
8             while (i < n1) and (j < n2):
9                 if L1[i] <= L2[j]:
10                    R.append(L1[i])
11                    i = i+1
12                else:
13                    R.append(L2[j])
14                    j = j+1
15
16            for i in range(i, n1):
17                R.append(L1[i])
18            for j in range(j, n2):
19                R.append(L2[j])
20
21            return R
```

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

```
def mergesort(L):

    if len(L) <= 1:
        return L

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

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

***Solution 5* (What is it? – *3 points*)**

1. Values of L after calling `what(`l`,` x`)`:
   (a) L `= [1, 3, 4, 5, 6, 7]`
   (b) L `= [1, 1, 1, 2, 2, 4, 5, 5, 5]`
   (c) L `= [1, 1, 1, 2, 2, 3, 3, 3, 3, 4]`
   (d) L `= [1, 3, 5, 7, 9]`
2. The function `what(`L`,` x`)` deletes all the values of x in the sorted list L.