Reserved Cells

Ex. 1	
Ex. 2	
Ex. 3	
Ex. 4	
Ex. 5	
Ex. 6	
Tot.	

Algorithms and Programming

18 September 2019

Part I: Theory

Register Number ____________ Family Name ____________________ First Name ____________________

Course: ○ 01*OGDLP* 10 credit ○ 02*OGDLM* 12 credit

No books or notes are allowed. Solve exercises directly within the reserved space. Additional sheets are accepted only when strictly necessary. Available time: 60 minutes.

1. **(2.0 points)**
   Given the following sequence of integers stored in an array:

   3 1 2 4 5 9 0 0 1 3 4 8 7 3 2

   sort it in ascending order using counting sort.

   Show the content of arrays *A*, *B* and *C* and all relevant intermediate steps on the array *C*.

   Show how counting sort can be implemented in C language, and specify its asymptotic complexity. Motivate the conclusions intuitively.

```
     0  1  2  3  4  5  6  7  8  9 10 11 12
    ---------------------------------------
A:   3  1  2  4  5  9  0  0  1  3  4  8  7
C1: 0  0  0  0  0  0  0  0  0  0  0
C2: 2  2  1  2  2  1  0  1  1  1  0
C3: 2  4  5  7  9 10 10 11 12 13 13
    2  4  5  7  9 10 10 10 12 13 13  - B  0  0  0  0  0  0  0  0  0  0  7  0  0
    2  4  5  7  9 10 10 10 11 13 13  - B  0  0  0  0  0  0  0  0  0  0  7  8  0
    2  4  5  7  8 10 10 10 11 13 13  - B  0  0  0  0  0  0  0  0  4  0  7  8  0
    2  4  5  6  8 10 10 10 11 13 13  - B  0  0  0  0  0  0  3  0  4  0  7  8  0
    2  3  5  6  8 10 10 10 11 13 13  - B  0  0  0  1  0  0  3  0  4  0  7  8  0
    1  3  5  6  8 10 10 10 11 13 13  - B  0  0  0  1  0  0  3  0  4  0  7  8  0
    0  3  5  6  8 10 10 10 11 13 13  - B  0  0  0  1  0  0  3  0  4  0  7  8  0
    0  3  5  6  8 10 10 10 11 12 13  - B  0  0  0  1  0  0  3  0  4  0  7  8  9
    0  3  5  6  8  9 10 10 11 12 13  - B  0  0  0  1  0  0  3  0  4  5  7  8  9
    0  3  5  6  7  9 10 10 11 12 13  - B  0  0  0  1  0  0  3  4  4  5  7  8  9
    0  3  4  6  7  9 10 10 11 12 13  - B  0  0  0  1  2  0  3  4  4  5  7  8  9
    0  2  4  6  7  9 10 10 11 12 13  - B  0  0  1  1  2  0  3  4  4  5  7  8  9
    0  2  4  5  7  9 10 10 11 12 13  - B  0  0  1  1  2  3  3  4  4  5  7  8  9
B : 0  0  1  1  2  3  3  4  4  5  7  8  9
```

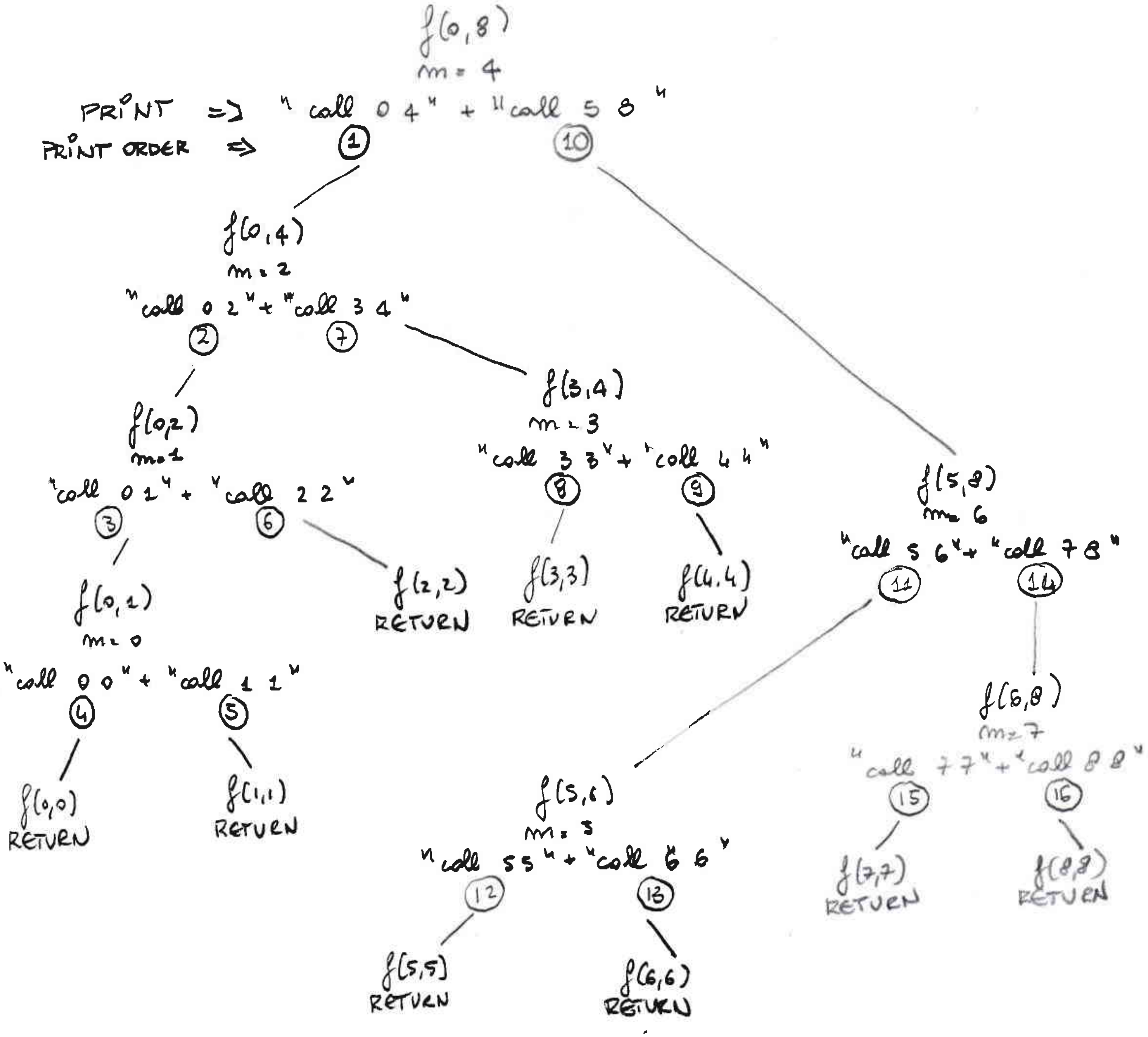
```
for(i=0; i<k; i++) C[i]=0;
for(i=0; i<n; i++) C[A[i]]++;
for(i=1; i<k; i++) C[i]+=C[i-1];
for(i=n-1; i>=0; i--) {
    B[C[A[i]]-1] = A[i];
    C[A[i]]--;
}
for(i=0; i<n; i++) A[i]=B[i];
```

T(n)= O(n+k)

2. (2.0 **points**)
Given the following program:

```
#include <stdio.h>

void f (int l, int r) {
  int m;

  if (l>=r) {
    return;
  }

  m = (l+r) / 2;
  fprintf (stdout, "call %d-%d\n", l, m);
  f (l, m);
  fprintf (stdout, "call %d-%d\n", m+1, r);
  f (m+1, r);

  return;
}

int main () {
  f (0, 8);
}
```

draw the recursive tree generated by function `f`. For each recursive call, report the values of all parameters and the exact output generated by the procedure.



3. (2.0 **points**)

10 **credit course** (01*OGDLP*)

Consider a binary tree with 13-nodes. Its visits return the following sequences:

pre-order:	*A*	*B*	*C*	*E*	*H*	*I*	*F*	*J*	*L*	*D*	*G*	*K*	*M*
in-order:	*A*	*H*	*E*	*I*	*C*	*J*	*L*	*F*	*B*	*G*	*M*	*K*	*D*
post-order:	*H*	*I*	*E*	*L*	*J*	*F*	*C*	*M*	*K*	*G*	*D*	*B*	*A*

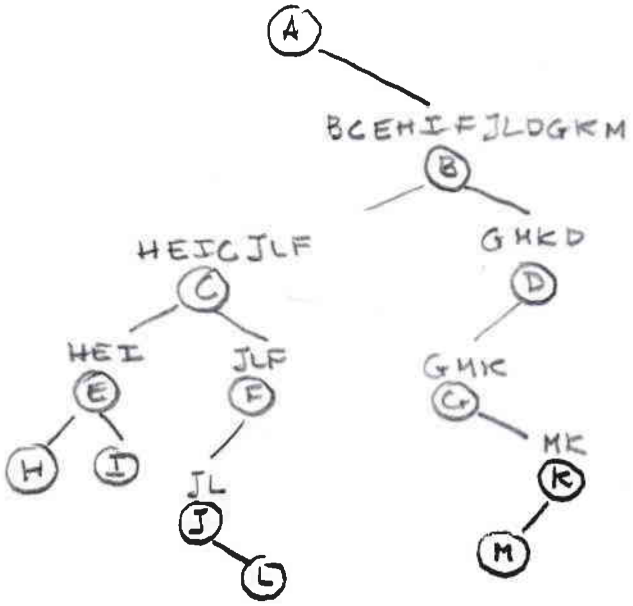
Draw the original binary tree.

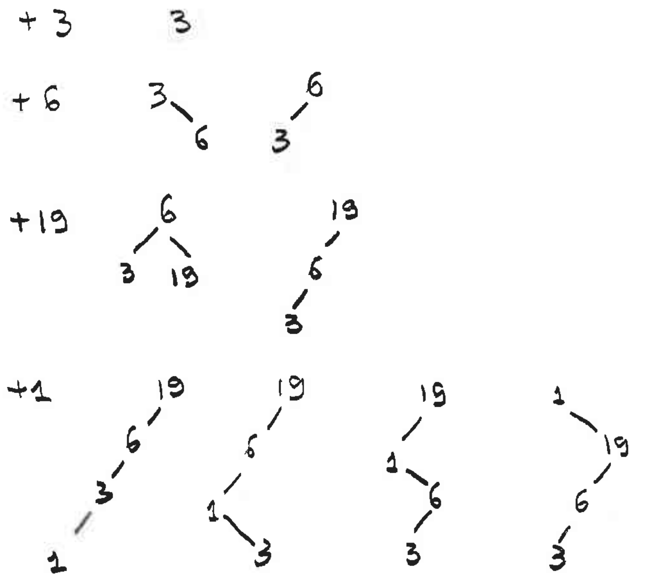
12 **credit course** (02*OGDLM*)

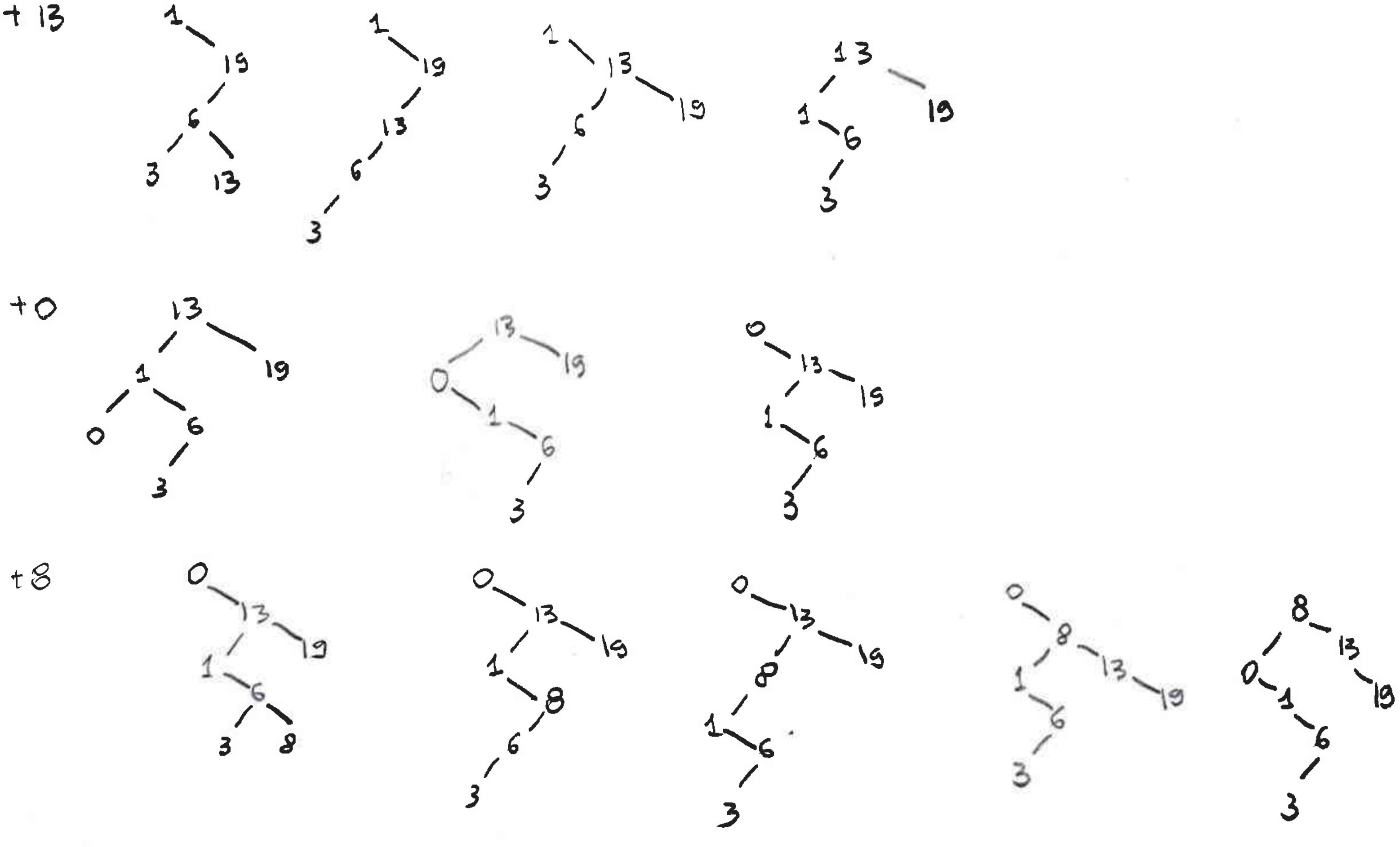
Given an initially empty BST perform the following sequence of insertions on the BST **root**:

3 6 19 1 13 0 8

Report all relevant intermediate steps.







4. **(2.0 points)**
Given the sequence of integers:

31 123 19 101 13 9 42 7 20 88 54 33

draw the final configuration of an initially empty hash table of size 23 where insertion of the previous sequence occurs. Assume open addressing with quadratic probing with $c_1 = 1$ and $c_2 = 1$. Show all relevant intermediate steps.

1	2	3	4	5	6	7	8	9	10	11		13		15		17		19		21		23		25	
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
											12		14		16		18		20		22		24		26

$h(k) = [(k \% 23) + c_1 i + c_2 i^2] \% 23$

i i^2

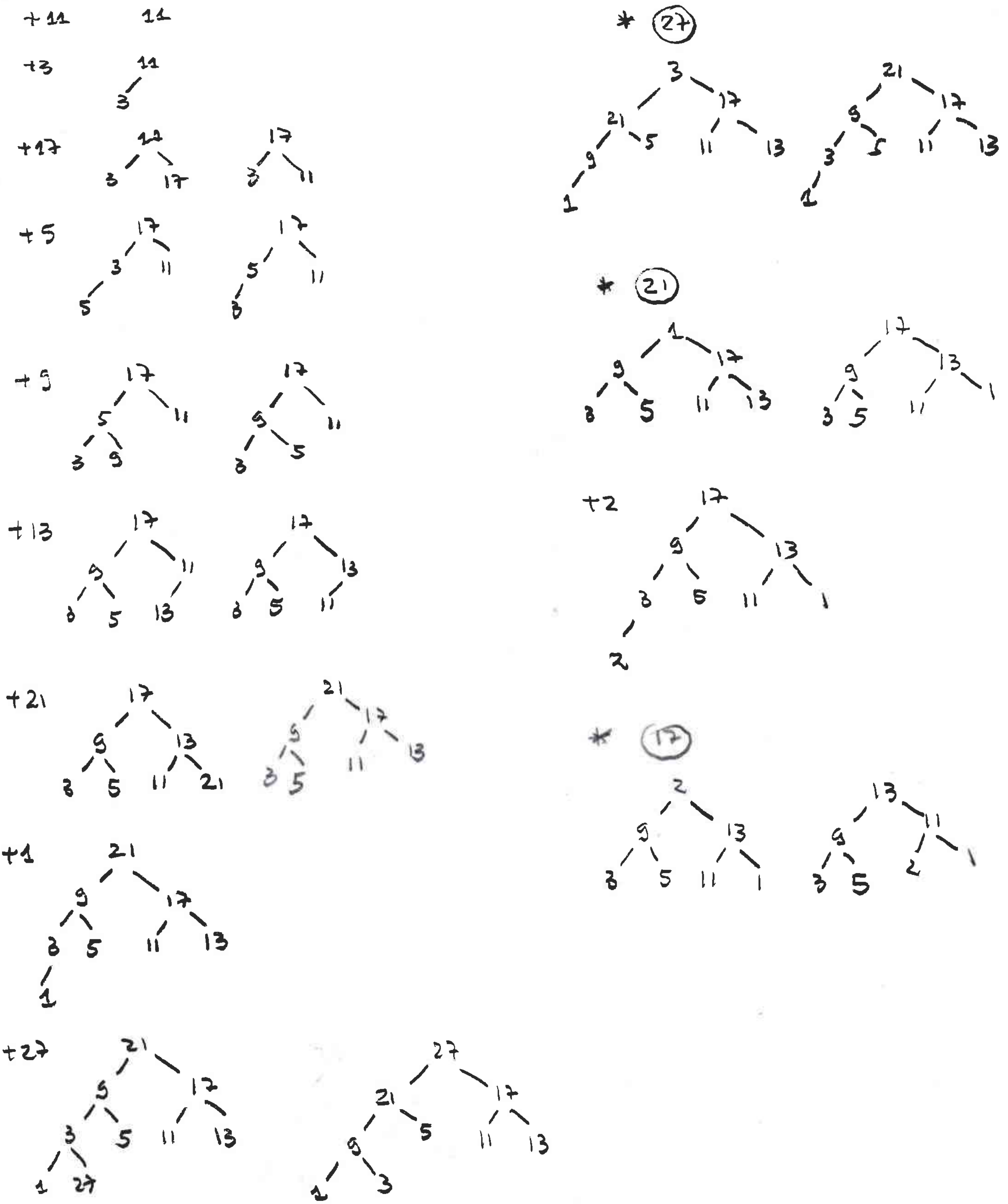
	i=0	i=1	i=2
31	8		
123	8 ☆	10	
19	19		
101	9		
13	13		
9	9 ☆	11	
42	19 ☆	21	
7	7		
20	20		
88	19 ☆	21 ☆	2
54	8 ☆	10 ☆	14
33	10	12	

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
		88					7	31	101	123	9	33	13	54					19	20	42	

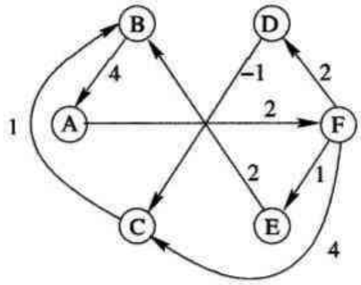
5. **(2.0 points)**
Suppose to have an initially empty priority queue implemented with a maximum heap. Given the following sequence of integers and $\star$ characters:

$$11 \quad 3 \quad 17 \quad 5 \quad 9 \quad 13 \quad 21 \quad 1 \quad 27 \quad \star \quad \star \quad 2 \quad \star$$

where each integer corresponds to an insertion into the priority queue and each character $\star$ corresponds to an extraction, show the priority queue after each operation and return the sequence of values extracted.



6. **(2.0 points)**
Given the following directed and weighted graph, find all shortest paths connecting node *A* with all the other nodes resorting to Bellman-Ford's algorithm.



If necessary, consider nodes and edges in alphabetical order.

Would Dijkstra's algorithm find the same result? Justify your answer.

LESSICOGRAPHIC ORDER

AF	2
BA	4
CB	1
DC	-1
EB	2
FC	4
FD	2
FE	1

6 VERTICES => 5 ITERATIONS

	0	1	2	3	4
A	0	0	0	0	0
B	∞	∞	7 → 5	4	4
C	∞	8 → 6	3	3	3
D	∞	4	4	4	4
E	∞	3	3	3	3
F	∞	2	2	2	2

STOP ✓

DIJKSTRA WOULD NOT WORK CORRECTLY (1 NEGATIVE WEIGHT EDGE)

BELLMAN FORD IS ALL RIGHT (NO NEGATIVE WEIGHT CYCLES)