INPUT : $n \geq 1$, $a_1, \ldots, a_n$

OUTPUT : MODIFIED LIST IN INCREASING ORDER.

1.) SET TOP TO n
2.) REPEAT UNTIL TOP < 2
3.) FIND INDEX imax OF MAX IN UNSORTED PART
4.) EXCHANGE a_{imax} WITH a_{TOP}
5.) SET TOP TO (TOP − 1)
6.) END loop
7.) STOP

THIS CAN ONLY BE CONSIDERED A FIRST DRAFT SINCE SOME OPERATIONS ARE NON PRIMITIVE. (STEPS 3 & 4).

WE REFINE STEP 3 BY RECALLING THE FIND-LARGEST ALGORITHM.

3.1) SET max TO a_1
3.2) SET imax TO 1
3.3) SET j TO 2
3.4) REPEAT UNTIL j > TOP
3.5) IF a_j > max
3.6) SET max TO a_j
3.7) SET imax TO j
3.8) SET j TO j+1
3.9) END loop

TO REFINE STEP 4 WE MUST INTRODUCE A NEW VARIABLE.

4.1) SET TEMP TO A_{imax}
4.2) SET A_{imax} TO A_{TOP}
4.3) SET A_{TOP} TO TEMP.

EX. $A_{imax} = 5$, $A_{TOP} = 3$, TEMP = ?

STEP	A_{imax}	A_{TOP}	TEMP
4.1	5	3	5
4.2	3	3	5
4.3	3	5	5

THE PROCESS OF REWRITING AN ALGORITHM WITH SUCCESSIVELY MORE DETAIL IS CALLED STEPWISE REFINEMENT, OR TOP DOWN DESIGN

BOTTOM UP DESIGN IS THE PROCESS OF PUTTING SIMPLE ALGORITHMS TOGETHER LIKE BLOCKS, TO FORM A COMPLICATED ALGORITHM.

ULTIMATELY ALGORITHMS MUST BE REFINED (OR BE REFINABLE) TO THE POINT WHERE ALL INSTRUCTIONS ARE PRIMITIVE.

TO ANALYSE THE TIME EFFICIENCY OF SELECTION SORT WE TAKE OUR BASIC WORK UNIT TO BE THE COMPARISON STEP (3.5). NOTE THIS IS THE SAME AS COUNTING THE NUMBER OF TIMES LOOP 3.4 – 3.9 IS EXECUTED.

(WE CONSIDER 2, 3.4, ETC.. TO BE PERIPHERAL TASKS.)

THIS LOOP IS CONTROLED BY j WHICH GOES FROM 2 TO TOP, AND SO EXECUTES (TOP−1) TIMES. BUT STEP (3) IS ITSELF INSIDE A LOOP CONTROLED BY TOP, WHICH GOES FROM n DOWN TO 2. THUS

TOP = n → (n−1) COMPARISONS
TOP = n−1 → (n−2) "
⋮
TOP = 2 → 1 COMPARISON

IN TOTAL THERE ARE

$$(n-1)+(n-2)+\cdots+2+1 = \frac{n(n-1)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$$

COMPARISONS PERFORMED.

IF WE HAD COUNTER PERIPHERAL TASKS THIS WOULD AFFECT THE COEFFICIENTS OF n^2 AND n, AND PERHAPS ADD A CONSTANT TERM, BUT IN ANY CASE, THE RUNNING TIME OF SELECTION SORT is $\Theta(n^2)$.

NOTE THAT THERE IS NO DISTINCTION HERE BETWEEN BEST CASE, WORST CASE, AND AVERAGE CASE. SELECTION SORT DOES THE SAME NUMBER OF COMPARISONS FOR ANY LIST OF LENGTH n.

BINARY SEARCH

RECALL THAT SEQUENTIAL SEARCH WAS $\Theta(1)$ IN BEST CASE AND $\Theta(n)$ IN WORST AND AVERAGE CASES.

AS WE SHALL SEE, BINARY SEARCH IS MUCH FASTER, BUT IT REQUIRES THAT THE LIST BE SORTED INITIALLY.

DESCRIPTION:

WE SET TWO MARKERS, L (LEFT) AND R (RIGHT), INITIALLY OUTSIDE THE (SORTED) LIST. WE FIND THE ELEMENT 'HALFWAY' BETWEEN THE TWO MARKERS AND COMPARE IT TO TARGET. IF IT IS LESS THAN TARGET WE MOVE L IN, IF IT IS LARGER THAN TARGET WE MOVE R IN. WE STOP WHEN EITHER WE FIND TARGET, OR THE TWO MARKERS CROSS.

EX. N=10

2 3 4 7 10 11 15 20 25 28

T = 11, 3 COMPARISONS.

EX. SAME LIST, T = 3, 2 COMPARISONS

EX. T = 7, 4 COMPARISONS

EX T = 8, 4 COMPARISONS.

IN ALL CASES WE DO (FAR) FEWER THAN 10 COMPARISONS.

INPUT: $n \geq 1$, $a_1, \ldots, a_n$ (SORTED, DISTINCT), T
OUTPUT: INDEX m FOR WHICH $a_m = T$, OR 0 IF NOT FOUND

1.) SET L TO 1, R TO n, FOUND TO FALSE
2.) REPEAT UNTIL FOUND OR R < L
3.) SET m TO $\lfloor \frac{R+L}{2} \rfloor$
4.) IF $T = a_m$
5.) SET FOUND TO TRUE
6.) ELSE IF $T < a_m$
7.) SET R TO m − 1
8.) ELSE
9.) SET L TO m+1
10.) END loop
11.) IF NOT FOUND
12.) SET M TO 0
13.) PRINT m
14.) STOP.

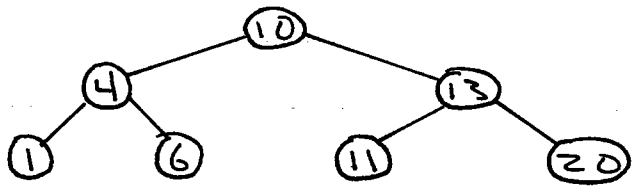
NOTE: THE FLOOR OF X, DENOTED $\lfloor x \rfloor$ IS THE GREATEST INTEGER WHICH IS LESS THAN OR EQUAL TO X. SIMILARLY THE CEILING OF X, DENOTED $\lceil x \rceil$, IS THE LEAST INTEGER WHICH IS GREATER THAN OR EQUAL TO X.

EX. $\lfloor 3.5 \rfloor = 3$, $\lceil 3.5 \rceil = 4$

AS WITH SEQUENTIAL SEARCH, WE WILL TAKE AS OUR BASIC UNIT OF WORK, THE COMPARISON OF A LIST ELEMENT WITH TARGET. (I.E. STEP 6 OR 4)

TO FACILITATE COUNTING THE NUMBER OF COMPARISONS DONE BY BINARY SEARCH, IT IS HELPFULL TO DIPICT ITS OPERATION BY A <u>BINARY SEARCH TREE</u>.

<u>EX.</u> 1 4 6 10 11 13 20 <u># COMPARISONS</u>



1
2
3

START WITH THE 'MIDPOINT' 10, THEN BRANCH TO THE MIDPOINTS OF THE LEFT AND RIGHT SUBLISTS, ETC.

THE DEPTH AT WHICH THE TARGET IS LOCATED IS THE NUMBER OF COMPARISONS BINARY SEARCH DOES TO FIND IT.

IF TARGET IS NOT IN THE LIST, THEN THE NUMBER OF COMPARISONS IS AT WORST THE MAXIMUM DEPTH OF THE TREE.