Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa

L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

CSI2110 Fall 2018

Final Exam

Prof. Lucia Moura and Prof. Robert Laganiere

December 10, 14:00 – 3 hours

40 points (40% of the final mark)

LAST NAME: SOLUTION OF CURRENT VERSION

First Name: AND MARKED UPDATES I AM MAKING

Student Number: _______________

Signature ____

Instructions :

Closed book exam. No calculators allowed. Please fill all multiple choice questions on scantron paper. There are 17 pages in this exam. In all the questions, when asked for a big-Oh, please provide the best possible value. All the logarithms are given in base 2.

QUESTION	MARKS OBTAINED
Q1-15 multiple choice	/15
Q16	/ 2
Q17	/ 2
Q18	/ 6
Q19	/ 4
Q20	/ 3
Q21	/ 2
Q22	/ 1
Q23	/ 2
Q24	/ 3
TOTAL	/40

Question 1 [1 point]
Consider the following piece of code:

```
int mycode(int n) {

   int i, j, k = 0;

   for (i  = n/2; i <= n; i++)

      for (j = 2; j <= n; j = j * 2)

         k = k + 2;

   return k;

 }
```

The time complexity of mycode(n) is

A)	$\Theta(n^2)$
[✓ B)]	$\Theta(n \log n)$
C)	$\Theta(n^3)$
D)	$\Theta(n^3 \log n)$
E)	None of the above

see changes

Question 2 [1 point]
Consider the following piece of code that uses an undirected graph G implemented with the adjacency lists data structure.

```
int count=0;

for each (v: G.vertices()) {  // run through all vertices v of the graph
          for each (e: G.incidentEdges(v)) // run through all edges incident to v
                    count++;
}
```

If n is the number of vertices and m is the number of edges, the function that most precisely describes the running time of this piece of code is:

A)	$O(n^3)$
B)	$O(n^2)$
C)	$O(n*m)$
[✓ D)]	$O(n+m)$
E)	None of the above

Question 3 [1 point]

Which of the first 4 statements is FALSE?

A) $2^{\log n}$ is $O(2^n)$ ✓

B) n^2 is $O(2^n)$ ✓

C) $2^{\log n}$ is $O(n^2)$ ✓

D) n^2+2^n is $\Omega(n)$ ✓

E) None of the above is false.

Question 4 [1 point]

For each of the following data structures: Heap, AVL tree, Hash Table, associate the Abstract Data Type that it implements:

	Heap	AVL tree	Hash table
A)	Priority queue	Map	Map
B)	Priority queue	Priority queue	Map
C)	Map	Map	Queue
D)	Queue	Priority queue	Map
E)	None of the above.		

Question 5 [1 point]

Which of the following alternatives is a true completion of the phrase:

The worst case running time of searching for a key in an AVL tree is (i) _________ but the worst case running time of searching for a key in a binary search tree is (ii) __________

A) (i) $\Theta(\log n)$ (ii) $\Theta(n)$

B) (i) $\Theta(\log n)$ (ii) $\Theta(n \log n)$

C) (i) $\Theta(n)$ (ii) $\Theta(\log n)$

D) (i) $\Theta(n \log n)$ (ii) $\Theta(n)$

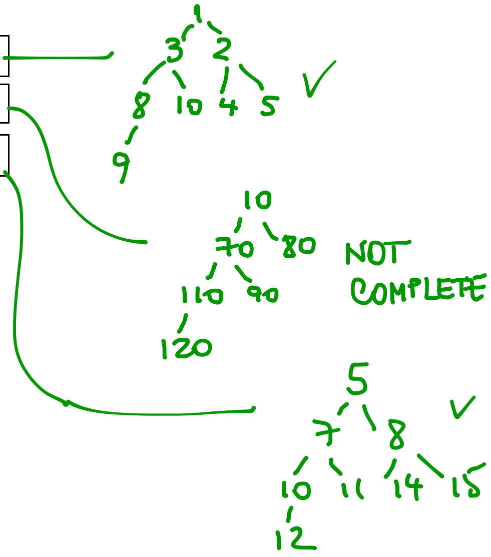
E) None of the above.

Question 6 [1 point] The worst case running times of the following sort algorithms are:

	Insertion sort	Mergesort	Quicksort
A)	Θ(n log n)	Θ(n log n)	Θ(n^2)
B)	Θ(n^2)	Θ(n^2)	Θ(n log n)
C)	Θ(n^2)	Θ(n log n)	Θ(n log n)
[✓ D)]	[✓ Θ(n^2)]	[✓ Θ(n log n)]	[✓ Θ(n^2)]
E)	None of the above.		

Question 7 [1 point] We use an array representation for a binary tree which stores keys:

I)	1	3	2	8	10	4	5	9
II)	10	70	80	110	90	-	-	120
III)	5	7	8	10	11	14	15	12



Which of the above represents a **min heap**?

A) I only

B) I and II only

[✓ C) I and III only]

D) II and III only

E) None of the above is the correct answer.

Question 8 [1 point]

Consider the **max heap** that uses the following array a[ ]=[10, 5, 3, 1, 4, 2] to store its elements. Which is the array after one removeMax() operation?

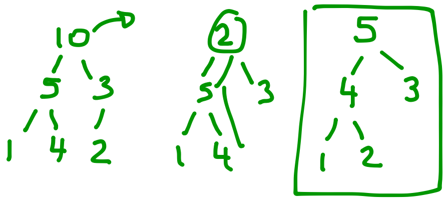
A) [5, 3, 1, 4, 2,-]

B) [2, 5, 3, 1, 4,-]

[✓ C) [5, 4, 3, 1, 2,-]]

D) [5, 2, 3, 1, 4,-]

E) [5, 4, 3, 2, 1,-]



Question 9 [1 point]

Consider the **max heap** that uses the following array a[ ]=[10, 5, 3, 1, 4, 2] to store its elements. Which is the array after inserting key 9 into the heap?

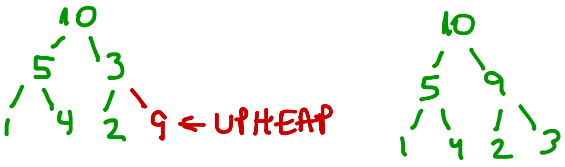
A) [10,5,3,1,4,2,9]

B) [10,9,5,1,4,2,3]

C) [10,9,1,4,5,2,3]

[✓ D) [10,5,9,1,4,2,3]]

E) none of the above



Question 10 [1 point] Consider the following Hash table where insertions are done using the hash function h(k)= k mod 7, and collisions are resolved with **quadratic probing**.

0	1	2	3	4	5	6
	15	**1**	**8**		**5**	

$h(8) = 1$

$\rightarrow (1+1) \bmod 7 = 2$

$(1+4) \bmod 7 = 5$

$(1+16) \bmod 7 = 3$

Regarding the ordering of insertion in this table, which answer is CORRECT?

A) Key 1 was the last key to be inserted.
[✓ B) Key 8 was the last key to be inserted.]
C) Key 15 was the last key to be inserted.
D) It is impossible to determine which was the last key among 8 and 5.
E) None of the above is correct.

Question 11 [1 point] Consider the following Hash table where insertions are done using the hash function h(k)= k mod 7, and collisions are resolved with **linear probing**.

0	1	2	3	4	5	6
	15	**1**	**8**	2	**5**	

2

What is the **average number of probes** A for searching an existing key in this table?

A) A=1
[✓ B) 1 < A < 2]
C) A= 2
D) A >2
E) None of the above is correct.

Key	# probes
15	1
1	2
8	3 +
5	1
	7

AVERAGE = 7/4

Question 12
Suppose you **insert element 2** in the table given in Question 11, still using linear probing. After this, you search for element 3. How many table positions must be probed until you conclude element 3 is not in the table?

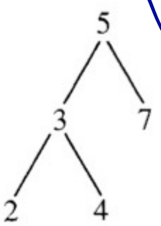
A) 2 B) 3 [✓ C) 4] D) 5 E) 6 or more

Question 13 [1point] Consider an algorithm that printed the nodes of the tree below in the order: 5 3 7 2 4. Which algorithm this may be?

WILL CHANGE TO

24375

NO QUESTION ABOUT TREES



A) Pre-order traversal of the tree.
B) Post-order traversal of the tree.
C) Depth-first search starting in node 5
D) Breadth-first search starting in node 5
E) None of the above is correct.

ANSWER WILL BECOME Post order B

Question 14 [1 point] Consider the torus graph, a graph where every vertex has exactly 4 adjacent vertices. The graph has **N vertices** and is represented using **adjacency lists data structure**. Taking into account the number of edges in this type of graph, the time complexity of running Depth-First Search (DFS) as a function of **N** is

A) $\Theta(\log N)$
B) $\Theta(N)$
C) $\Theta(N \log N)$
D) $\Theta(N^2)$
E) None of the above is correct.

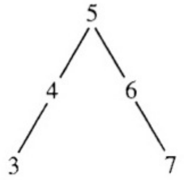
$m = \frac{4 \times N}{2} = 2N$

DFS is $\Theta(N+m) = \Theta(N+2N)$
$= \Theta(N)$

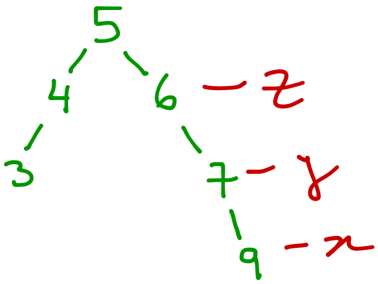
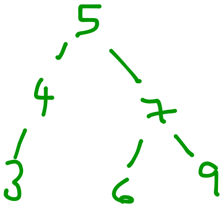
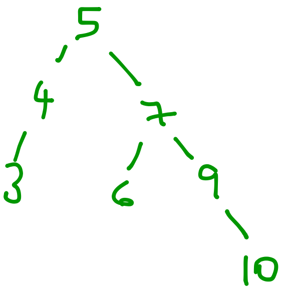
Question 15 [1 point] Consider procedure stableSort(String [] A, int i) which sorts a global array of strings with 4 letters by simply using the letter at position i of each string S as the key for sorting in increasing order of S[i]. Moreover, we know that the algorithm is **stable**, that is, if elements have the same key, their order in the output will be the same order they arrived in the input. Which of the following algorithms sorts the strings in array A in alphabetical order?

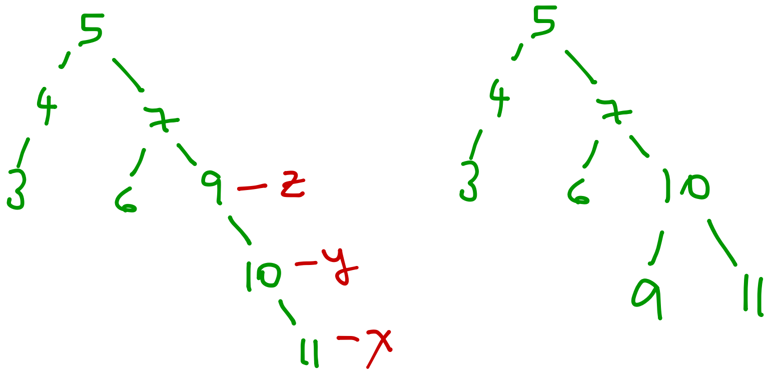
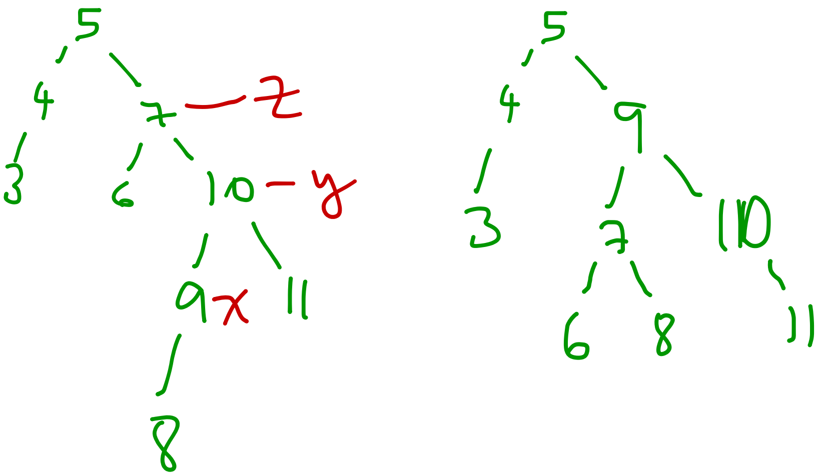
A) for (i=0 ; i<4 ; i++) stableSort(A,i)
B) for (i=3 ; i>=0 ; i--) stableSort(A,i)
C)
```
for (i=0 ; i<4 ; i++)
    for (j=0 ; j<i ; j++)
        stableSort(A,i)
```
D) for (i=0 ; i<=log n ; i++) stableSort(A,i)
E) None of the above.

Question 16 [2 marks] Consider the following AVL tree where dummy nodes exist but have not been drawn.

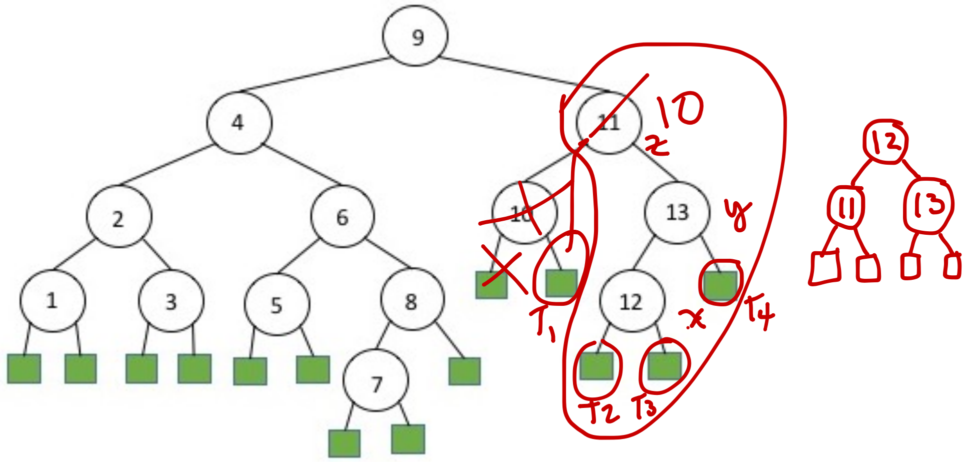


The following insertions are done, one at a time in this order, into the AVL tree above: Insert 9, Insert 10, Insert 11, Insert 8. Show the tree after each insertion:

After Inset 9	 
After Insert 9, Insert 10	

After Insert 9, Insert 10, Insert 11	 5 4 7 3 6 9 – z 10 – y 11 – x 5 4 7 3 6 10 9 11
After Insert 9, Insert 10, Insert 11, Insert 8.	 5 4 7 – z 3 6 10 – y 9 x 11 8 5 4 9 3 7 10 6 8 11

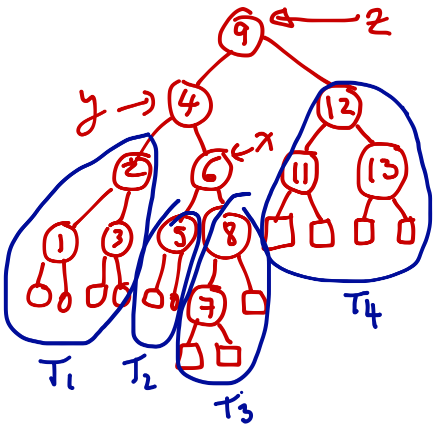
Question 17 [2 marks] Consider the following AVL tree.



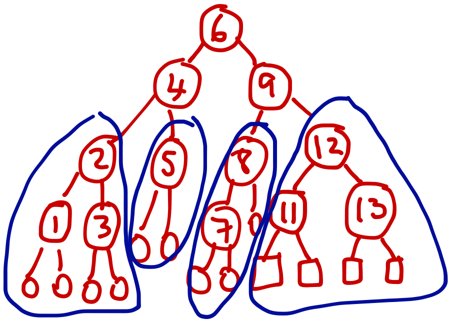
ADD LINE

Show the relevant steps (tree transformations) after the operation delete 11, assuming the first step in the deletion operations selects element 10 to substitute the deleted element 11. Whenever a rebalancing operation is performed, identify the relevant nodes (x, y, z) and sub-trees involved in the rebalancing operation (before and after the rebalancing takes place)

REBALANCING 1



REBALANCING 2



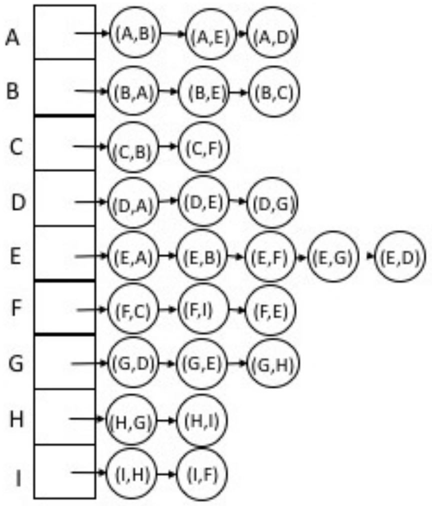
(page intentionally left blank)

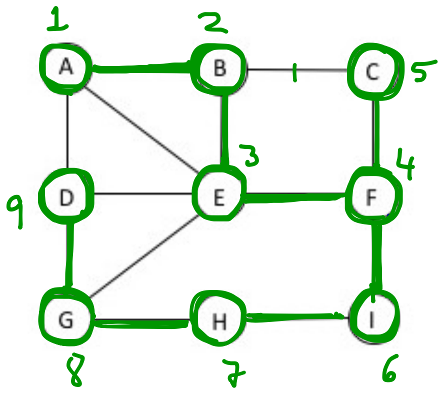
Question 18 [6 points=3+3] Consider the graph given below and consider the given adjacency list representation of the graph.

1)Taking into account the order of the vertices in each adjacency list, apply **depth-first search** starting from **vertex A.** Please draw on the graph:

a) The order in which the nodes are visited (mark the number beside the vertex)

b) Mark in the graph the edges that are part of the spanning tree computed by the algorithm (tree of "discovery" edges).

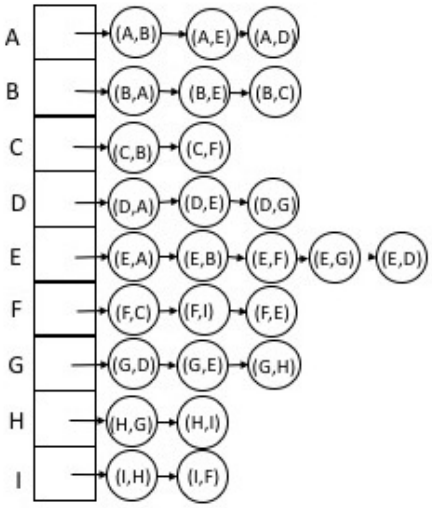


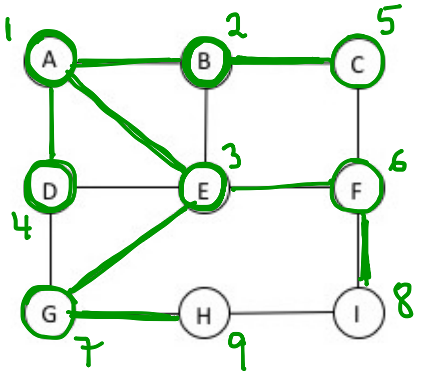


2.) Taking into account the order of the vertices in each adjacency list, apply **breadth-first search** starting from **vertex A.** Please draw on the graph:

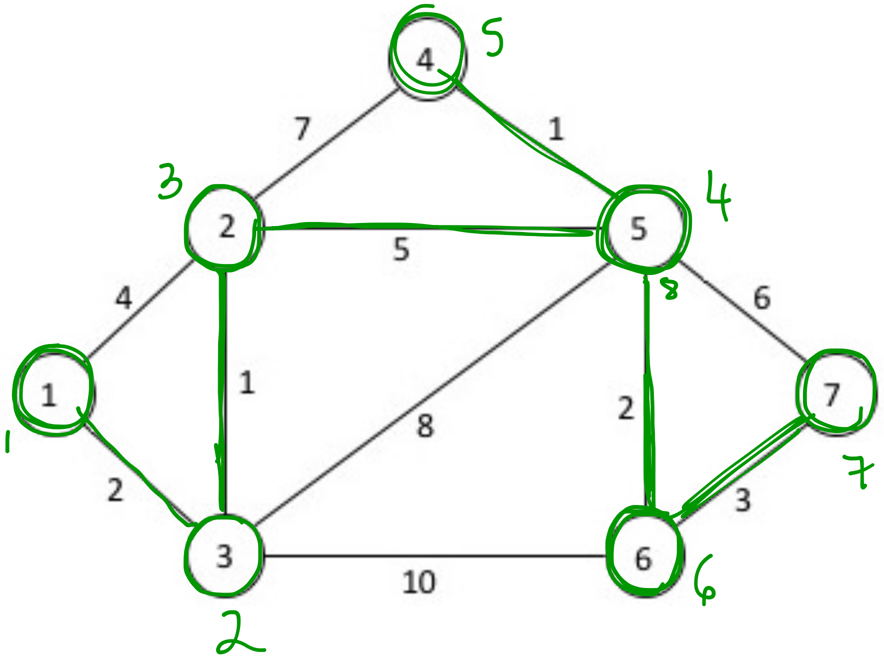
a) The order in which the nodes are visited (mark the order number beside the vertex)

b) Mark in the graph the edges that are part of the spanning tree computed by the algorithm (tree of "discovery" edges).





Question 19 [4 points]
Use Dijkstra's algorithm to obtain a tree of shortest paths for the graph below starting from vertex 1 :



Display below:

a) Vertices in the order they enter the "cloud" (tree of shortest paths),
b) Edges in the order they enter the tree of shortest paths (tree edges).
Give edges in the format (u,v); e.g. (1,3) for the edge that connects vertices 1 and 3.
c) Final array with distances **dist**, where **dist[v]** shows the distance between origin (vertex 1) and vertex v

Vertices (in order)
1
3
2
5
4
6
7

Edges (u,v) (in order)
(1,3)
(3,2)
(2,5)
(5,4)
(5,6)
(6,7)

	dist
1	0
2	3
3	2
4	8
5	8
6	10
7	13

Question 20 [3 points] Mergesort

Simulate a full execution of the in-place Mergesort algorithm for the given array below. Suppose that at the end of the "merge" step, the algorithm prints a line with the current contents of the full array.

Show each printout of this algorithm, and highlight the part of the array where the merge has been done. The number of blank arrays below may be more or less than the amount you need to show (use the back if more space is needed).

30	10	50	20	15	7	2	12
10	30	50	20	15	7	2	12
10	30	20	50	15	7	2	12
10	20	30	50	15	7	2	12
10	20	30	50	7	15	2	12
10	20	30	50	7	15	2	12
10	20	30	50	2	7	12	15
2	7	10	12	15	20	30	50

NOTE: I DECIDED TO LEAVE THE INPLACE VERSION FOR ENGLISH SECTION

Question 21 [2 points] The following program is an implementation of the Quicksort algorithm.

```
/* Sort the subarray data[low..high] inclusive */
public static void doQuickSort(int[] data, int low, int high) {

          if (low >= high) return;

          // beginning of Partition
          int left = low; int right = high-1; int pivotPos=high;

          int pivot= data[pivotPos];

          while (left <= right) {
               while (left <=right && data[left]<pivot) left++;

               while (left <=right && data[right]>pivot) right--;

               if (left <= right) {
                   temp=data[left]; data[right]=data[left]; data[right]=temp;
                   left++; right--;
               }
           }
           int pos=left;
           int temp=data[pos]; data[pos]=data[pivotPos]; data[pivotPos]=temp;
          // *** end of Partition ***

           doQuickSort(data, low, pos-1);
           doQuickSort(data, pos+1, high);
}
```

The most important part of the quicksort strategy for sorting is the partition of the array into two parts using a pivot element. The value of the chosen pivot determines the efficiency of the sorting. Consider the following array `data` in the call `doQuickSort(data,0,7)`:

18	35	12	20	15	7	2	50

L

WILL CHANGE THE NUMBERS SO MY GOOD PARTITION COMES FIRST.

a) Show what will be the state of the array at the end of the first partition (i.e. at the point indicated in the source code by the ***).

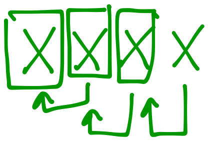
18	35	12	20	15	7	2	50

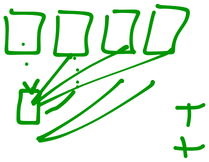
b) Now if you change the line `int left = low; int right = high-1; int pivotPos=high;`
by: `int left = low+1; int right = high; int pivotPos=low;`
and also change the line `int pos=left;` by `int pos=right;`
Will you get the same result? If not, draw the state of the new array at the same point ***

15	2		7	18	20	35	

↑ ↑ ↑ ↑ ↑ ↑ ↑

↑ R ↑ L





3
+2
+1
6

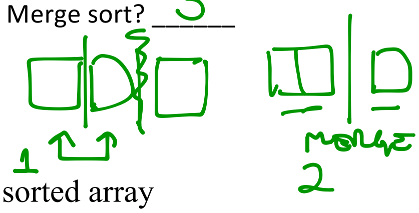
Question 22 [1 points]

Consider an array of four equal-value elements. [X,X,X,X]

A) How many comparisons will be performed if you try to sort this array using insertion sort? 3

B) How many comparisons will be performed if you try to sort this array using Selection sort? 6

C) How many merge operations will be performed if you try to sort this array using Merge sort? 3



.

Question 23 [2 points]

A programmer wrote a wrong implementation of a binary search method for a sorted array `data`. He/she tested the method and it finds the right value. However the method seems to be slower than expected.

the method

a) Examine the code and evaluate the worst-case complexity of the implemented algorithm. The algorithm runs in O(N)

b) Binary search in a sorted array should be O(log N), modify the code below such that it now correctly implements this algorithm. Note: you have the right to change a maximum of three lines in this code.

```
// search if a given integer `value' is in a sorted array `data' of size N>0
public int search(int value, int[] data, int N) {

  int inf=0, sup= N-1;
  int mid= (inf+sup)/2;

  while (data[mid]!=value && inf<sup) {

    if (data[mid]<value) {
      ~~inf++;~~ inf= mid+1;
    } else {
      ~~sup--;~~ Sup= mid-1;
    }

    mid= (inf+sup)/2;
  }

  if (data[mid]==value)
    return mid;
  else
    return -1;
}
```

JUSTIFICATION
(NOT REQUIRED)
At each iteration
the size considered
decreases by 1,
at the begining
(sup-inf)+1 = N
it will end when
this = 1.
(N-1) STEPS.

QUESTION THAT WILL ARISE:
CAN I USE BASIC DATA STRUCTURES?
EG. ARRAYLIST(E), LINKEDLIST(E), ETC

I WILL ADD A NOTE ABOUT THAT.

Question 24 [3 points]
Suppose you are given an array A of student names and grades (each grade is an integer number between 0 and 10). Write an algorithm to sort this array in increasing order of grades in time O(N), where N is the number of students.
At the end of the algorithm, the input array must be sorted.
Hint – implement the BucketSort method.
You may and should use other auxiliary memory to store the buckets and its contents.
Reminder: BucketSort creates an array of lists indexed by the values the key can take, insert each record on the list indexed by the value of its key, then uses these lists to produce a sorted output.
Write your algorithm in Java-like pseudocode or Java code.

```
public interface Student {
    public int grade(); // key used for sorting
    public String StudentName();
}
public void BucketSort (Student[] A) {
LinkedList<Student>[ ]  Buckets = new LinkedList<Student>[11];
[OPTIONALLY INITIALIZE IT WILL NULL]

 for i=0; i< A.lenght; i++) {
       st = A[i];
       grade = st.grade();
       if (grade<0 || grade>10) { print Error;
                                   return}
       if (Bucket[i]==null)
              Bucket[i] = new LinkedList<Student>;
       Bucket[i].insertLast(st);
   }
j=0;
for (b=0; b<11; b++) {
     for (Student s : Bucket[i]) {
           A[j] = s;
           j++;
     }
   }
}
```

(page intentionally left blank)