המרכז האקדמי לב

שנה"ל **תשע"ט**, סמסטר **ב'**, מועד **ב**
שאלון בחינה בקורס: מבנה נתונים ותוכניות א
מספר קורס: **150015**

Student ID #:_____

Campus: Lev

Instructor's Name: Dr. Stulman

Date of Exam: 16/8/2019 ט"ו באב, תשע"ט
Length of Exam (in minutes): 180
Materials Allowed During Exam: None, besides the enclosed formula sheet

This exam has 6 questions each worth 20 points. Answer any 5 of them. If you answer 6 some random 5 will be marked.

Answers must be written on answer sheets provided. Draft notes will not be checked.

1. **If for any reason you are not certain if you understand the intent of the instructor with regard to a question, you are to indicate at the beginning of your answer how you understand the question and answer accordingly. The instructor has the right to determine how many points your understanding and answer are worth.**
2. **You must return the test form with the notebook (in the case where there is a notebook). If the form is not attached the test will not be graded.**
3. **It is your responsibility to understand the institutional regulations regarding exams. Any deviation from those rules may lead to the exam being disallowed and other actions taken against student.**
4. **Please note that points will be taken off not only for mistakes but for irrelevant material in answer, lack of adequate explanation for an answer, an unclear and/or ambiguous answer.**

Good Luck!

Place an X on the question you **do not** wish to be marked:

1	2	3	4	5	6	Total

General comments:

1. When the exam mentions the term run-time without additional qualifications, the worst-case run-time is meant (in order of magnitude).
2. When the exam stipulates that additional memory cannot be used, you are allowed to use a fixed amount of additional memory.
3. In questions where you are required to write an algorithm, you can use algorithms taught in the lectures without re-writing them.

Question 1

A. (10 points): Prove or refute:

$$n^3 - \sqrt{n} = \Omega(n^2 * \sqrt{n})$$

B. (10 points): Let

$$g(n) = n^2 \quad , \quad f(n) = \begin{cases} n^2 & if\ n\ mod\ 2 = 0 \\ n & else \end{cases}$$

Prove or refute:

$$f(n) = \theta(g(n))$$

Question 2:

Below is a list of data structures:

1. An un-sorted singly-linked list
2. A sorted doubly-linked list
3. Min heap
4. Max heap

Complete the following table with the worst-case running times for each operation on each of the data structures:

Structure 4	Structure 3	Structure 2	Structure 1	Operation
****				Find(key)
			*	Insert(key)
		**		Delete(key)
	***			Maximum
				Minimum

Note: *Maximum* and *Minimum* do not remove the max/min elements from the structure; they merely return its value.

You must justify the marked locations

Justification for *

Justification for **

Justification for ***

Justification for ****

Question 3

A. (10 points) Prove or refute:
Given a binary search tree *T* and a value *j* (*j* is not necessarily in *T*), the node in *T* with key *k* closest to j (i.e. the minimal |k - j|) is on the search path for *j* in *T*.

Reminder: The *search path* for *j* in *T* is the series of nodes one visits in *T* when searching for *j*.

B. (10 points) Given an array of numbers of length *n* with $\lfloor n/logn \rfloor$ unsorted numbers followed by sorted numbers. Write, in pseudo-code, a linear time algorithm [i.e. *O(n)*] for sorting the array.
You must explain your algorithm and prove its running time.

Question 4:

Let *A[1..n]* be and array containing the numbers $a_1, \ldots a_n$ obtained by a *pre-order* traversal of a binary search tree *T* holding *n* distinct values.

A. (10 points): Assume that the root of *T* had a right son. Write – in pseudo-code – an algorithm with a running time of $O(logn)$ to find the index in the array containing the value of that right son of the root.

B. (10 points) Given that A=[40,30,20,10,35,80,50,60,70], draw the tree *T* from which A was derived.

Question 5:

A. (10 points): Build an AVL tree from the following values (left to right). You must draw the tree after each insertion, and name the rotation used every time it is required.

22, 10, 7, 5, 30, 17, 15, 14, 13, 12, 11, 18

B. (10 points): Delete from the tree you built in section A the following values (left to right). You must show each step of the deletion, and name the rotation used each time it is required.

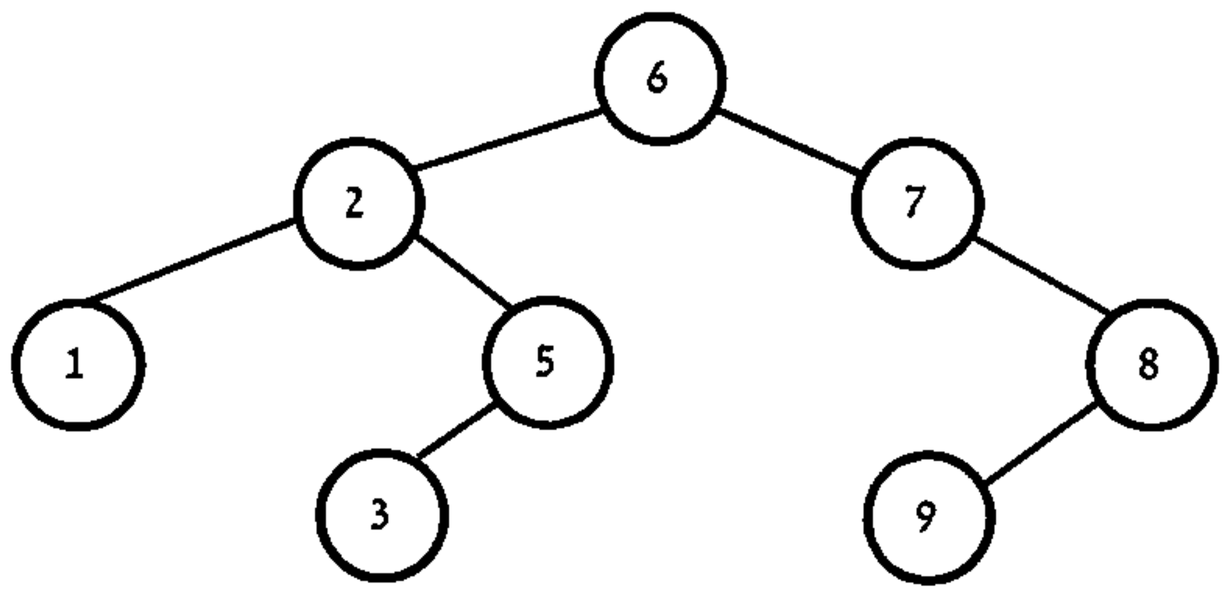
10, 22, 12, 15

Question 6:

Below you are provided with a code fragment that receives the root of a binary tree as input:

```
boolean Doit(Node *root)
{
    Stack S;
    Node *p=root;
    int y=0;
    while (!S.isEmpty() or p!=NULL)
    {
        while (p!=NULL)
        {
            S.push(p); p=p->left;
        }
        If (!S.isEmpty())
        {
            p=S.pop();
            if (p->value < y) return false;
            y=p->value;
            p=p->right;
        }
    }
    return true;
}
```

A. Given the following tree:



A1. (5 points) What is the result of the algorithm when run on the tree?

A2. (5 points) What will the stack contain when the algorithm terminates (when the above tree is provided as input).

B. (5 points): In general, what does the algorithm check? Justify.

C. (5 points): When given a tree with *n* elements, what is the running time of the algorithm? Justify.

Good Luck !!

Additional space for your answers:

Formula Sheet

Definition of O : Given two functions f(n), g(n): $\mathbb{N} \to \mathbb{R}^+$
We say that g(n) is O(f(n)) if there are positive constants n_0 and c such that
$g(n) \leq c*f(n)$ for all $n \geq n_0$

Definition of Ω : Given two functions f(n), g(n): $\mathbb{N} \to \mathbb{R}^+$
We say that g(n) is Ω (f(n)) if there are positive constants n_0 and c such that
$g(n) \geq c*f(n)$ for all $n \geq n_0$

Definition of Θ : Given two functions f(n), g(n): $\mathbb{N} \to \mathbb{R}^+$
We say that g(n) is Θ (f(n)) if there are positive constants n_0 , c_1, c_2 and c such that
$c_1 f(n) \leq g(n) \leq c_2 f(n)$ for all $n \geq n_0$

Definition of o : Given two functions f(n), g(n): $\mathbb{N} \to \mathbb{R}^+$
We say that g(n) is o(f(n)) if there are positive constants n_0 and c such that
$g(n) < c*f(n)$ for all $n \geq n_0$

Definition of ω : Given two functions f(n), g(n): $\mathbb{N} \to \mathbb{R}^+$
We say that g(n) is ω (f(n)) if there are positive constants n_0 and c such that
$g(n) > c*f(n)$ for all $n \geq n_0$

Arithmetic Series

$$\sum_{k=1}^{n} k = 1+2+\ldots+n = \frac{1}{2}n(n+1)$$

Geometric Series

$$\sum_{k=0}^{n} x^k = 1+x+x^2+\ldots+x^n = \frac{x^{n+1}-1}{x-1}$$

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x} \qquad |x|<1 \qquad \text{עבור}$$

Harmonic Series

$$H_n = 1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\ldots+\frac{1}{n} = \sum_{k=1}^{n}\frac{1}{k} = \ln n + O(1)$$

Series of Squares

$$\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6} \text{ :סדרת הריבועים}$$

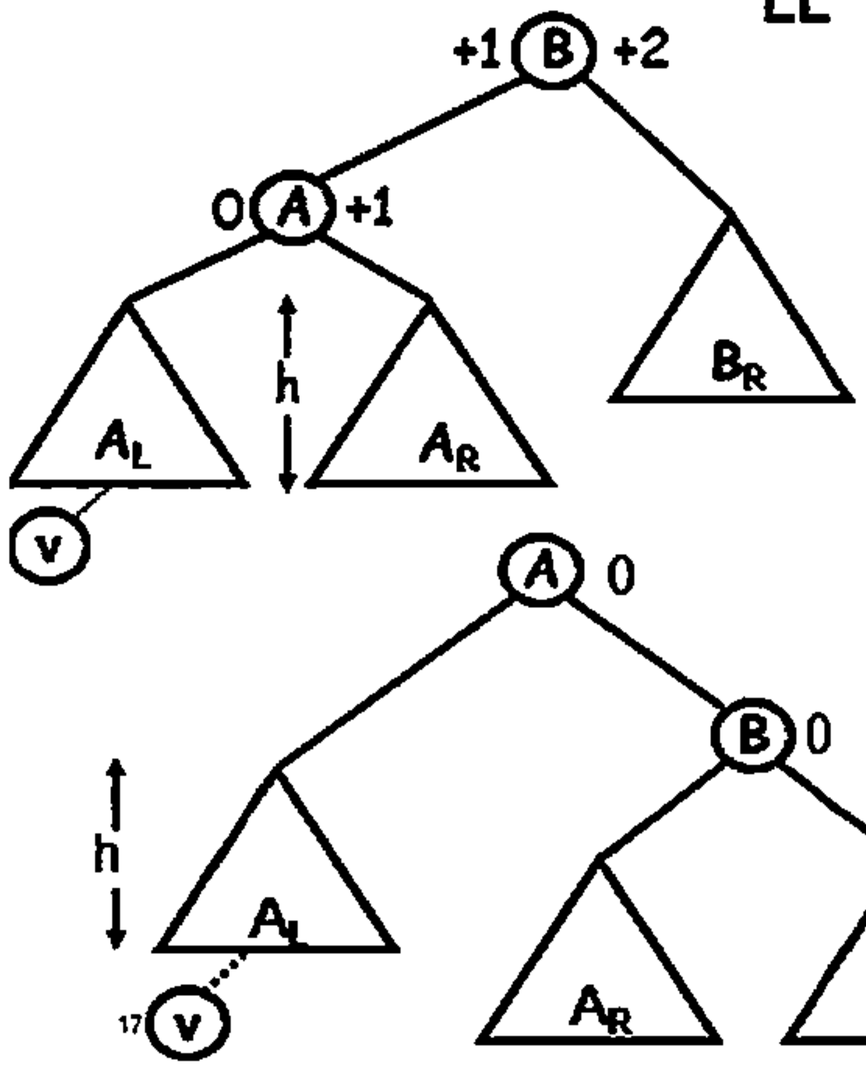
שינוי בסיס log: $\frac{\log_c a}{\log_c b} = \log_b a$

שינוי חזקה: $n^{\log_c a} = a^{\log_c n}$

כלל לופיטל: $\lim_{x\to\infty} \frac{f(x)}{g(x)} = \lim_{x\to\infty} \frac{f'(x)}{g'(x)}$

AVL

גלגול LL



<u>לפני הכנסת v</u>: גובה העץ הוא h+2.

הוכנס צומת v שהגדיל את גובה A_L ל-h +1.

מצד ימין של הצמתים מסומנים גורמי האיזון שהשתנו.

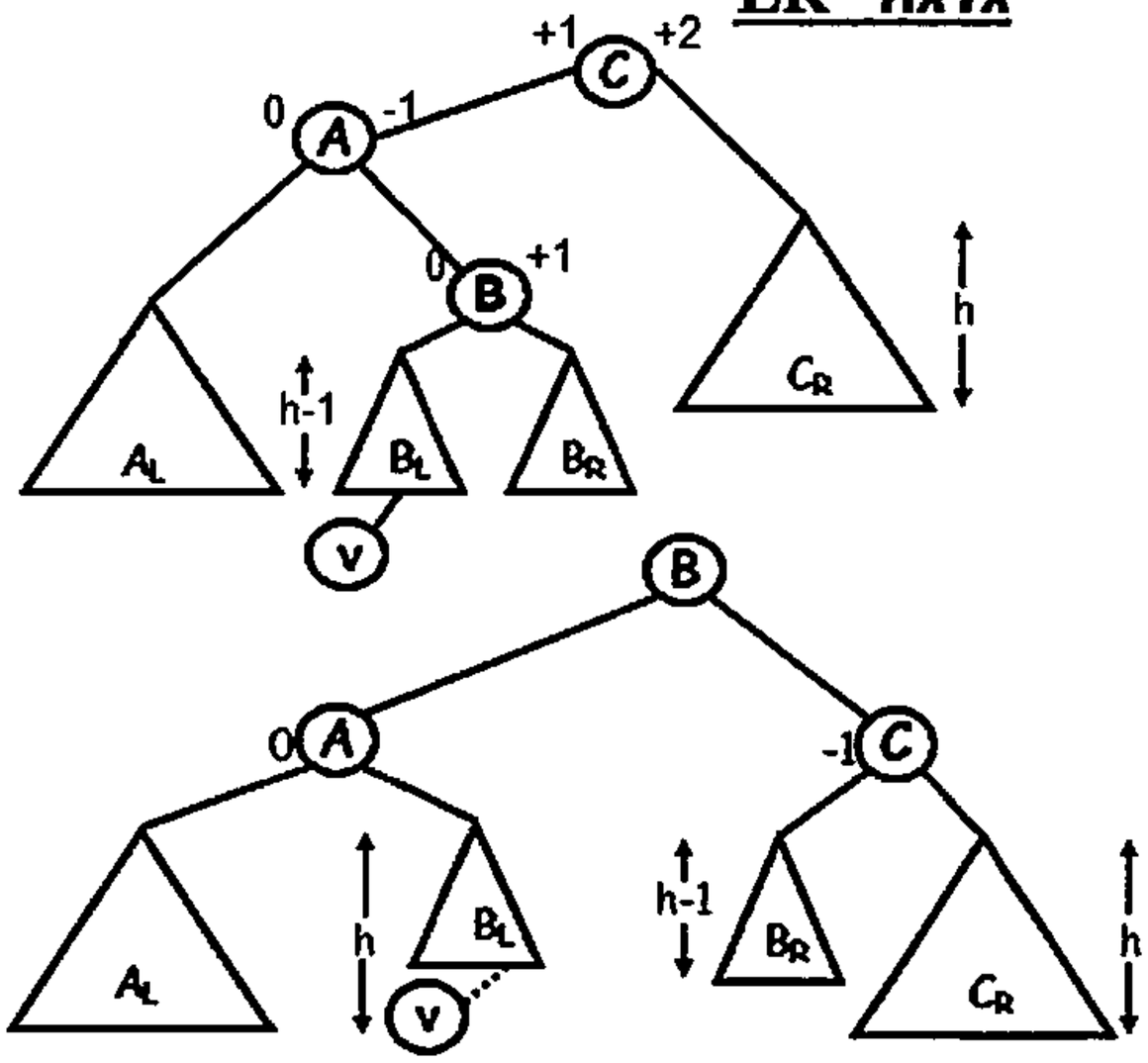
<u>גלגול LL</u> : יעביר את A לשורש

<u>אחרי הגלגול</u>:

גובה העץ לאחר הגלגול הוא h +2, כמו לפני ההכנסה. השורש מאוזן. שינוים O(1) מצביעים ולכן זמן הגלגול O(1).

AVL

גלגול LR



לפני הכנסת איבר v:

הוכנס איבר ל-B_L שגרם לו להעלות את גובהו ל-h.

גלגול LR:

גובה העץ אחרי הגלגול הוא h+2, כמו לפני ההכנסה.

שינוים O(1) מצביעים ולכן זמן הגלגול O(1).