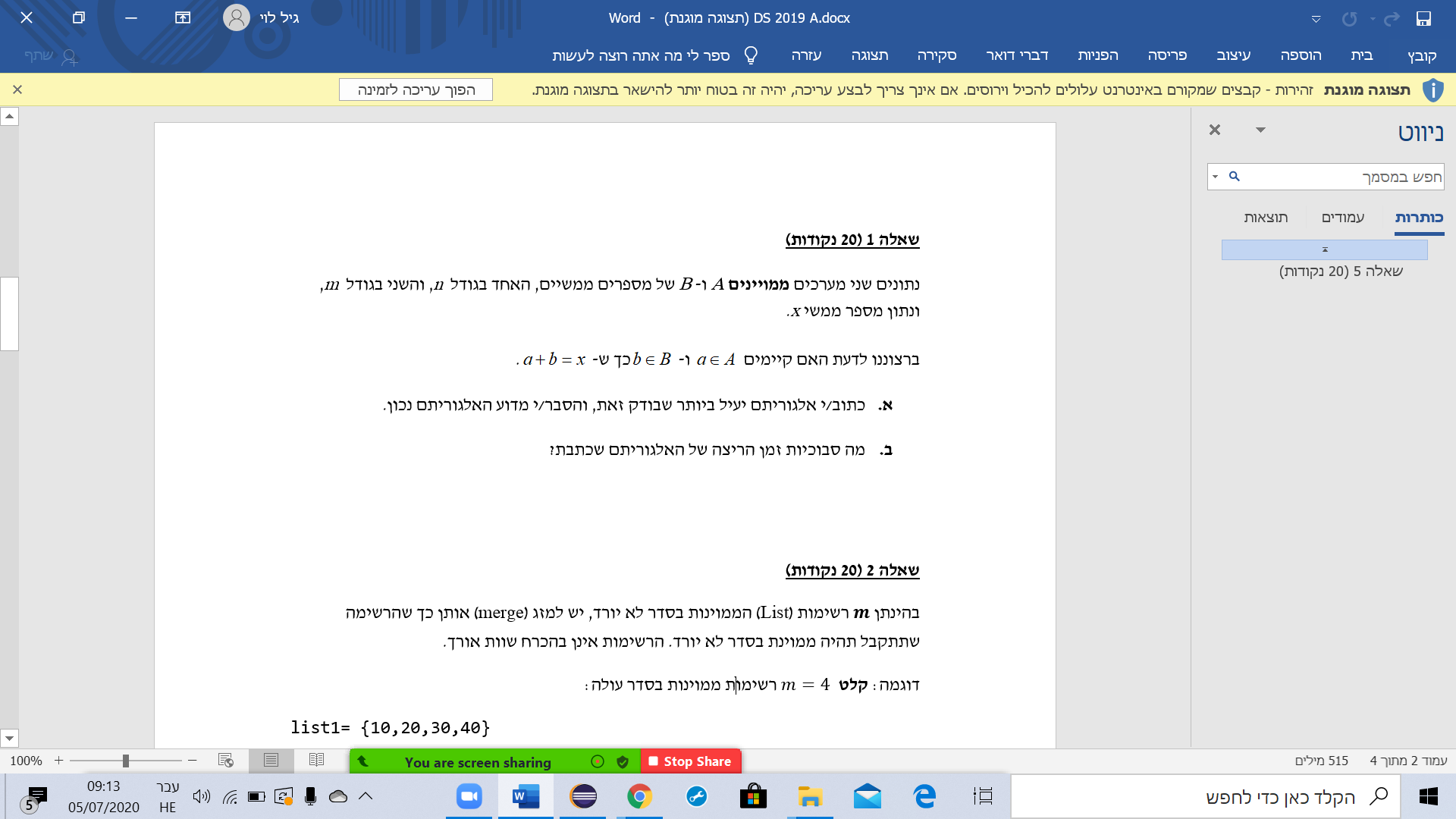
מרתון מבני נתונים - 5.7

מבחן 2019 סממסטר ב מועד א



פתרון:

א. רעיון: נחבר את האיבר הראשון ב A (הקטן ביותר ב A) עם האיבר האחרון ב B (הגדול ביותר ב B)

אם קיבלנו x - מצאנו את האיברים, אם קיבלנו תוצאה גדולה מ x. אז נרד לאיבר הקודם ב B (כי אם האיבר הכי קטן ב A ביחד עם האיבר הכי גדול ב B עוקף את התוצאה, כל שכן אם נחבר לאיבר הגדול ב B איבר יותר גדול מ A) ולכן האיבר הכי גדול ב B אינו פוטנציאלי. באותו אופן ובכיוון ההפוך, אם התוצאה קטנה יותר מ x נעבור לאיבר הבא ב A.

פסאודו קוד:

q1(A,B,x), |A|=n , |B|=m

a = 0

b = m - 1

while(a<n and b>=0)

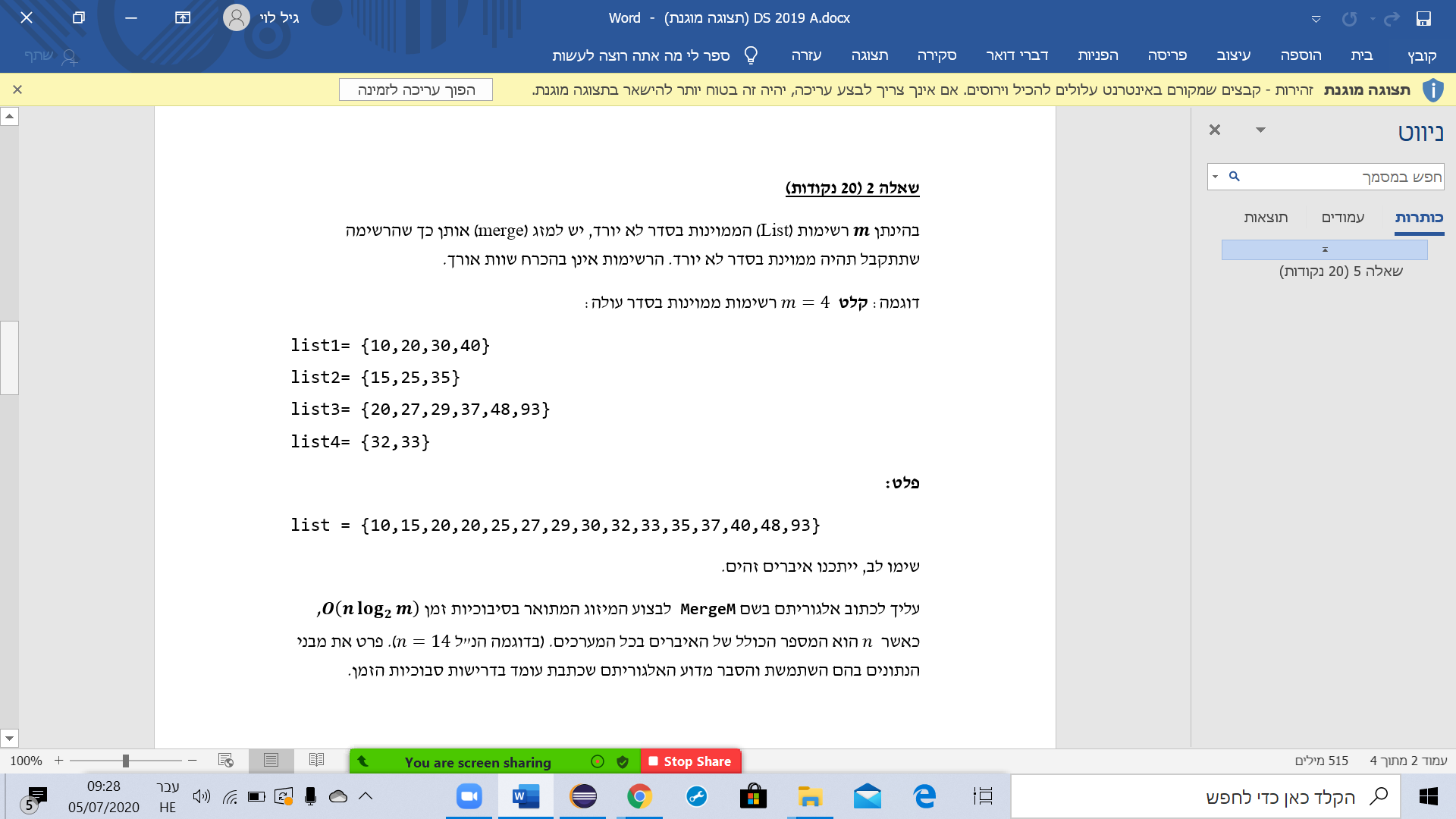
if(A[a]+B[b] = x) return true

else if(A[a]+B[b] < x) a++

else b--

return false

ב. סיבוכיות: בכל איטרציה של הלולאה או ש a מתקדם למעלה או ש b מתקדם למטה ולכן לכל היותר לאחר n+m איטרציות, הלולאה תסתיים, בנוסף, בכל איטרציה מבצעים פעולות ולכן סה"כ הסיבוכיות היא: .



פתרון:

נשתמש ב MinHeap בגודל m. נכניס אליו בתחלה את האיברים הראשונים מכל רשימה ביחד עם מספר הרשימה שלהם. מוציאים מ heap בכל שלב את המינימאלי ומכניסים לרשימת התשובה. ומכניסים ל heap את האיבר הבא ברשימה שממנה יצא האיבר המינימאלי.

פסאודו קוד:

q2(List<List> lst)

MinHeap h = init

for i = 0 to m-1

h.inset((lst[i].remove(0),i))

List ans = init

while(h is not empty)

x = h.extractMin()

ans.add(x.value)

h.insert(lst[x.numberOfList].remove(0))

return ans

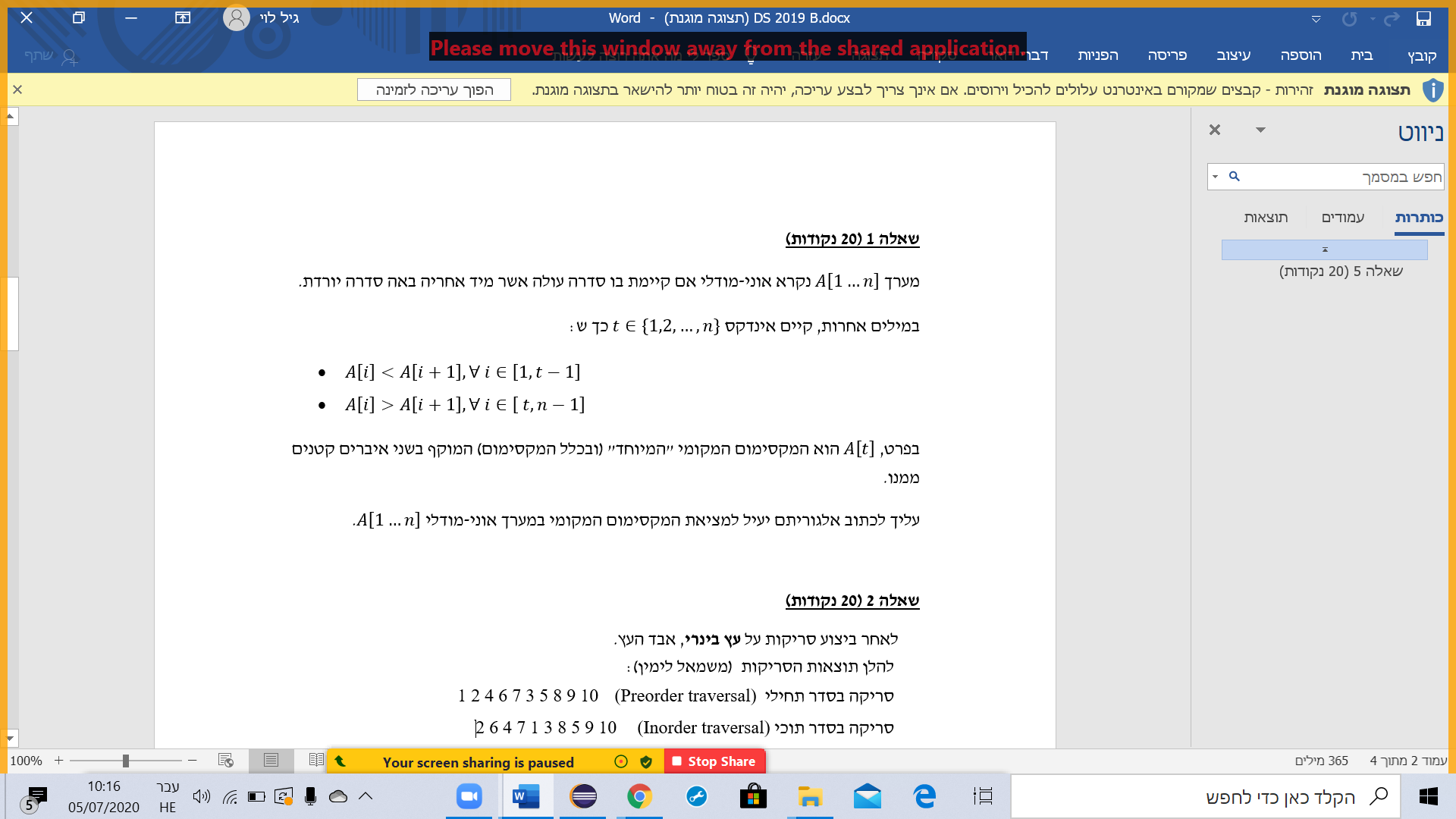
סיבוכיות של הכנסה והוצאת המינימום מ heap היא .

הכנסת כל האיברים ל heap היא .

מספר הפעמים שהלולאה תתבצע היא כי כל פעם מכניסים איבר ל heap עד שנגמרים האיברים ולכן כל אחד מ n האיברים יהיה ב heap פעם אחת.

בכל שלב מבצעים פעולות על ה heap. סה"כ:

מכיוון ש הסיבוכיות הכוללת תהיה .



פתרון:

נבצע חיפוש בינארי כאשר עבור כל איבר שנפגוש, נשאל האם 2 האיברים הסמוכים אליו קטנים ממנו, אם כן אז הוא האיבר המקסימאלי. אם לא, נבדוק אם האיבר הבא גדול מהאיבר הנוכחי אז אנו נמצאים בעלייה ולכן המקסימום יהיה בחצי הימני, אחרת אנחנו בירידה ולכן המקסימום יהיה בחצי השמאלי.

הנחה: המקסימום לא יהיה באיבר הראשון או האחרון.

q1(A)

low = 1, high = n

while(low <= high)

mid = (low+high)/2

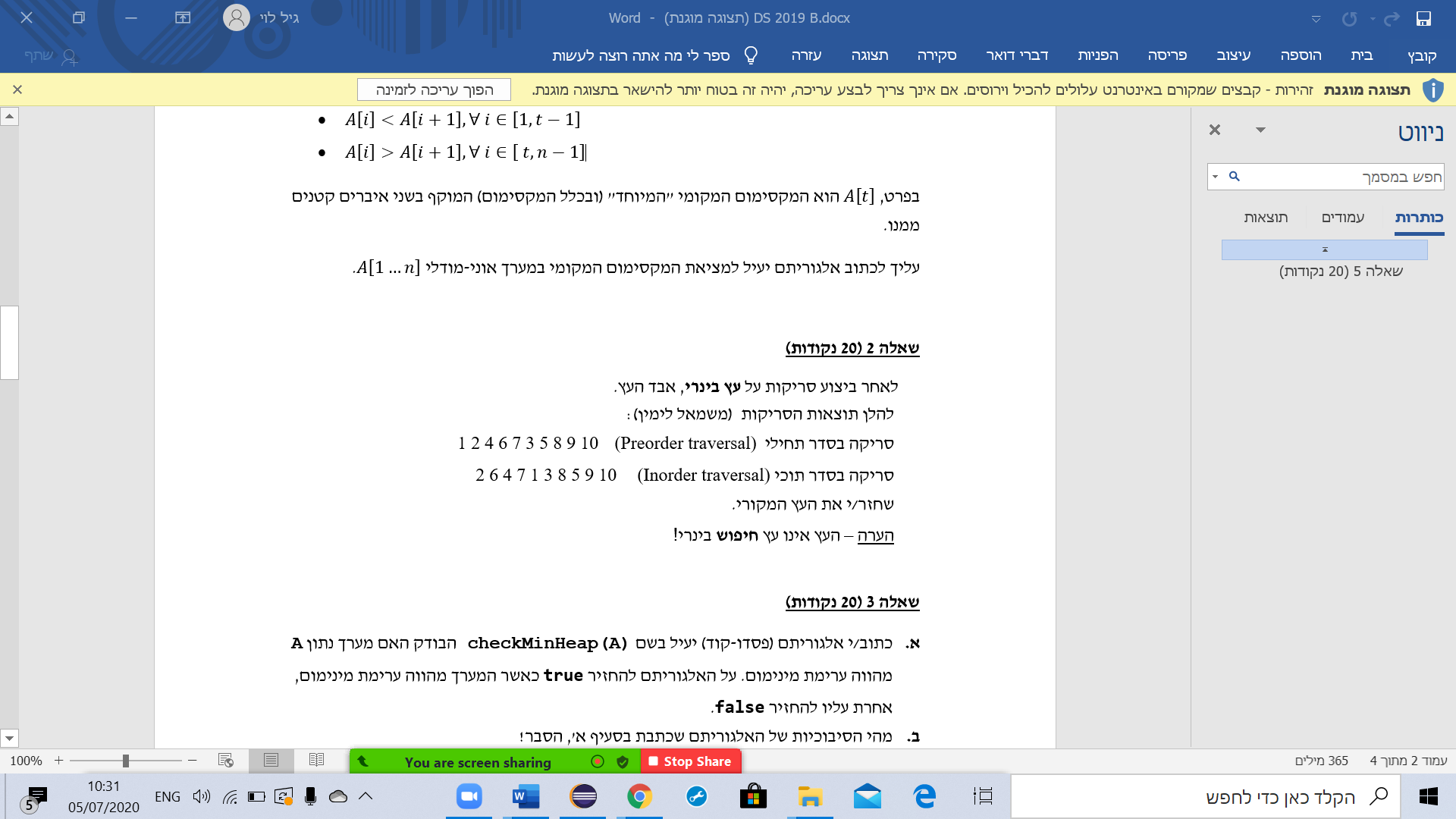
if(A[mid] > A[mid-1] and A[mid] > A[mid+1]) return mid

else if(A[mid] < A[mid+1]) low = mid+1

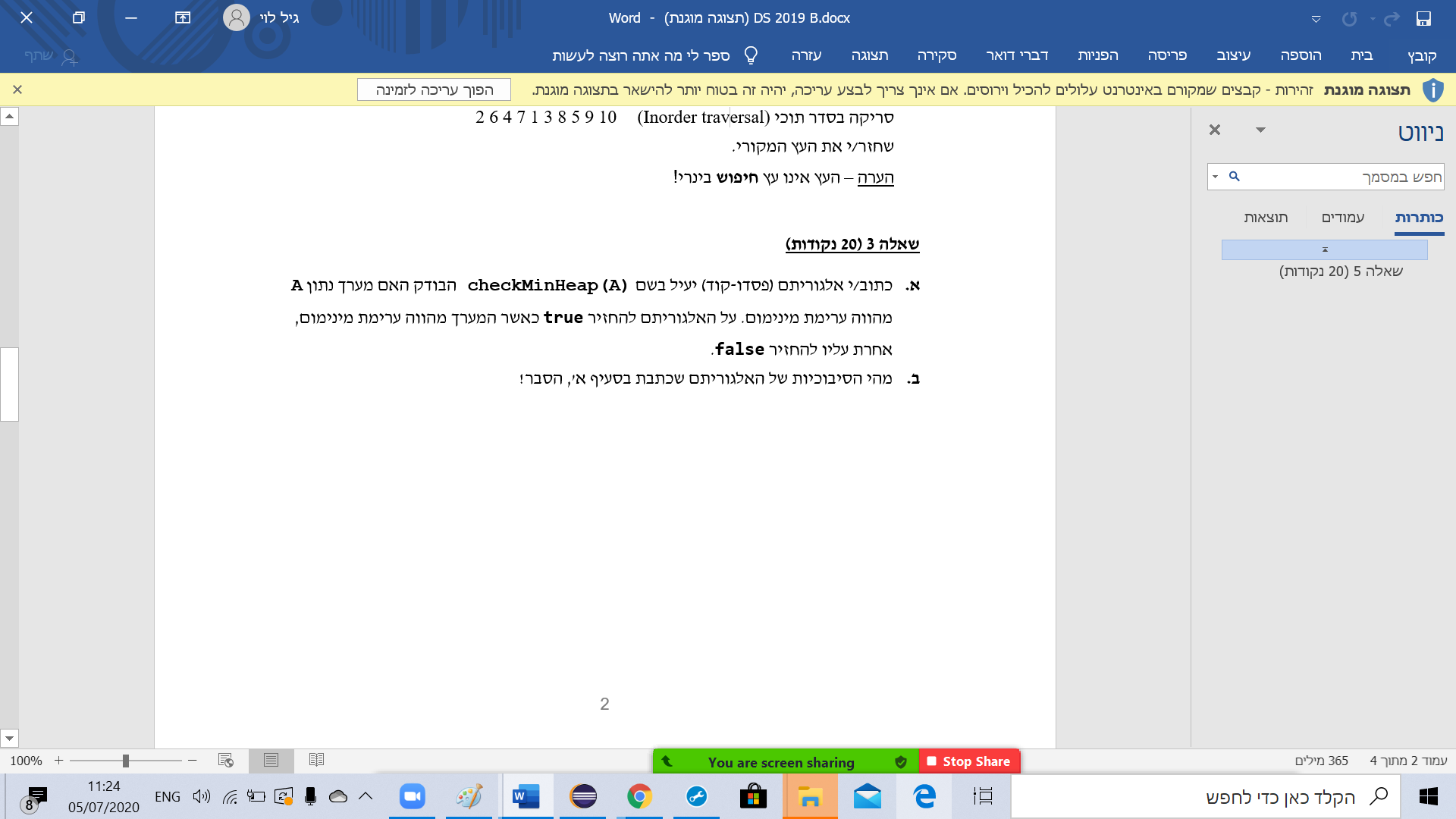
else high = mid-1

return -1

סיבוכיות כמו של חיפוש בינארי, כל פעם מחלקים ב 2 את המערך ובודקים רק חצי אחד. לכן סה"כ: .

מרתון מבני נתונים - 5

פתרון:



פתרון:

א. נעבור על המערך ונבדוק עבור כל תא המייצג אבא שיש לו בנים האם 2 הבנים גדולים בערכם ממנו.

checkMinHeap(A), |A|=n

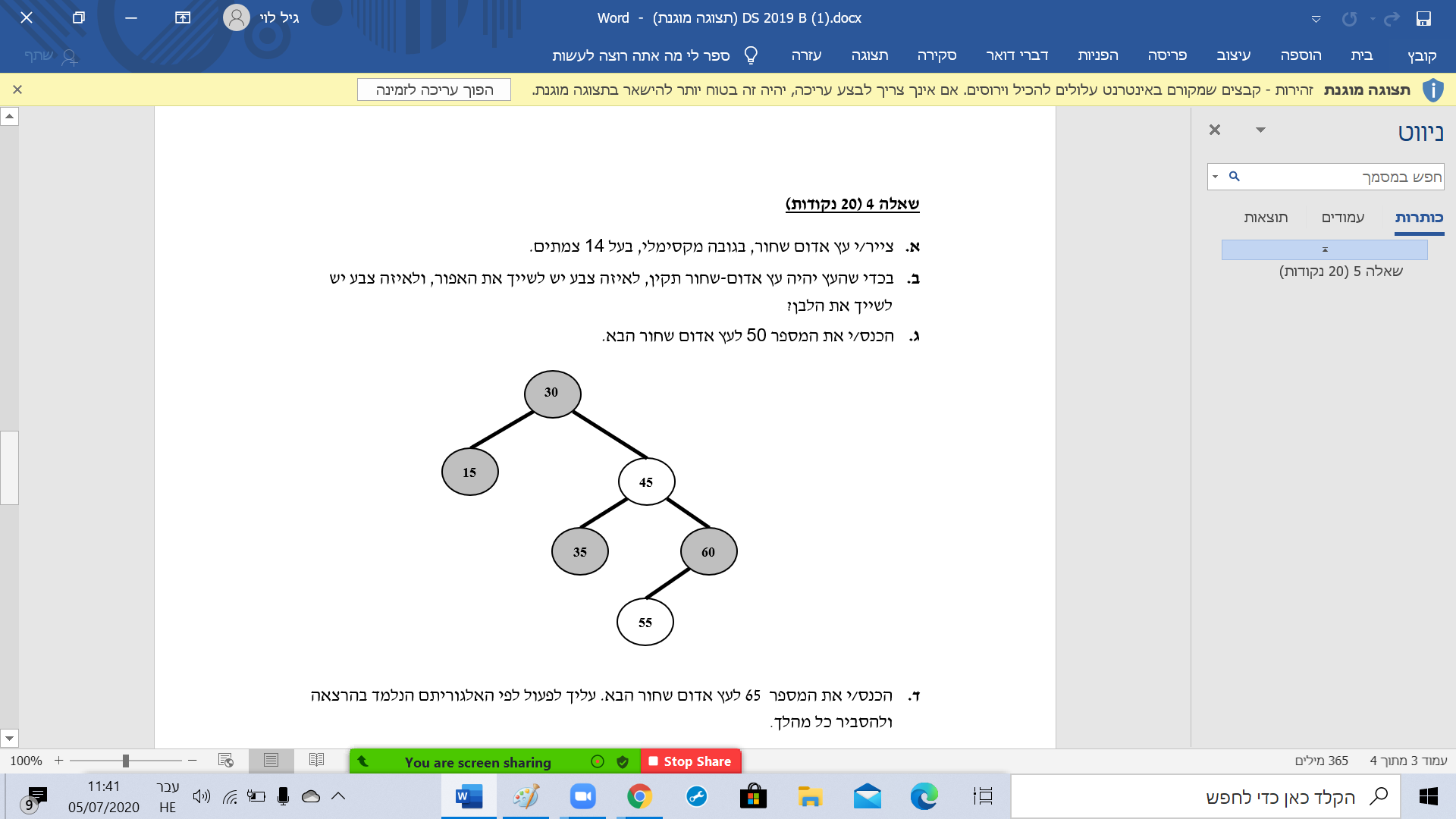
for i = 1 to n

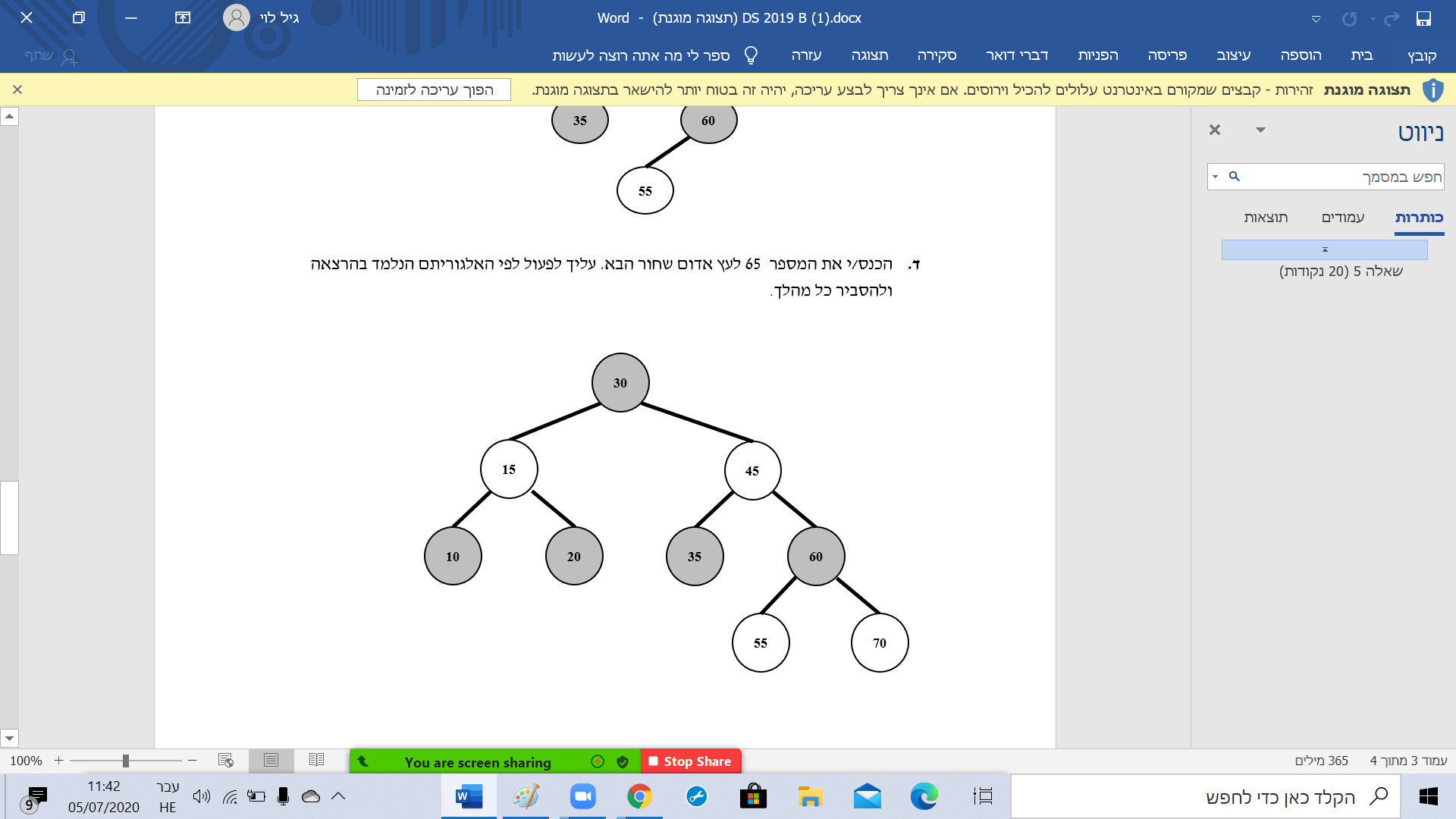
if(i\*2<=n and A[i\*2] < A[i]) return false

if(i\*2+1<=n and A[i\*2+1]<A[i]) return false

return true

ב. עוברים פעם אחת על המערך ומבצעים בכל פעם 2 פעולות. סה"כ: .



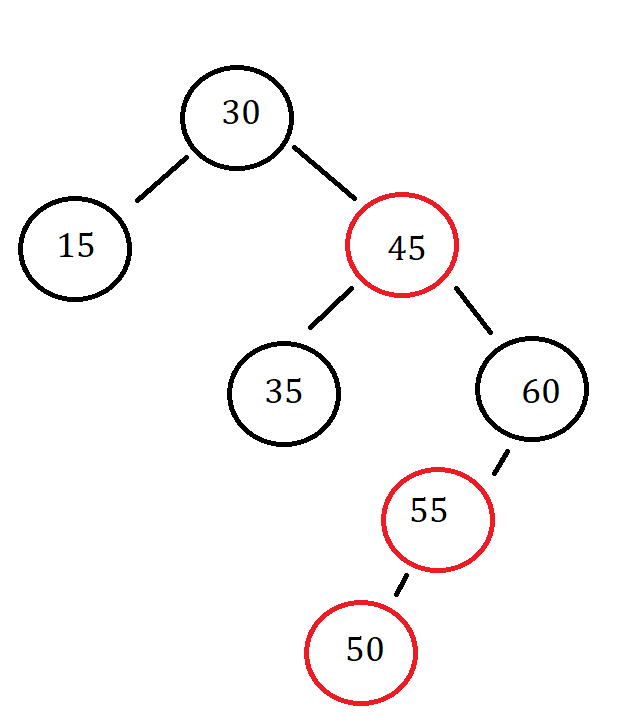


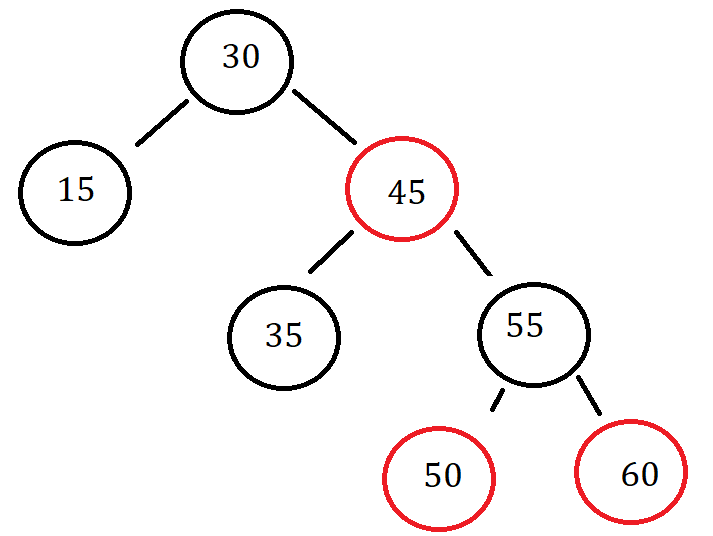
פתרון:

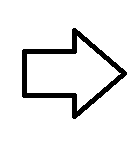
א. נוכיח שהגובה לא יכול להיות יותר מ 5. נניח בשלילה שהגובה הוא לפחות 6 ולכן קיים מסלול בעץ שיש בו לפחות 7 קודקודים. מכיוון שהשורש שחור ואסור 2 אדומים ברצף על המסלול. ישנם לפחות 4 שחורים במסלול. מכאן, בכל מסלול בעץ צריכים להיות לפחות 4 שחורים כדי לשמור על תקינות העץ. מכאן, אם כל מסלול בעץ מכיל לפחות 4 שחורים אז העץ שלם עד גובה 3. ולכן יהיו בן לפחות: 1+2+4+8=15 קודקודים בסתירה לנתון שיש בעץ רק 14 קודקודים. לכן הגובה יכול להיות לכל היותר 5.

נראה ציור של עץ תקין בגובה 5 ומזה נוכיח שהמקסימום הוא 5:

ב. השורש תמיד שחור ולכן האפור הוא השחור והלבן הוא האדום.

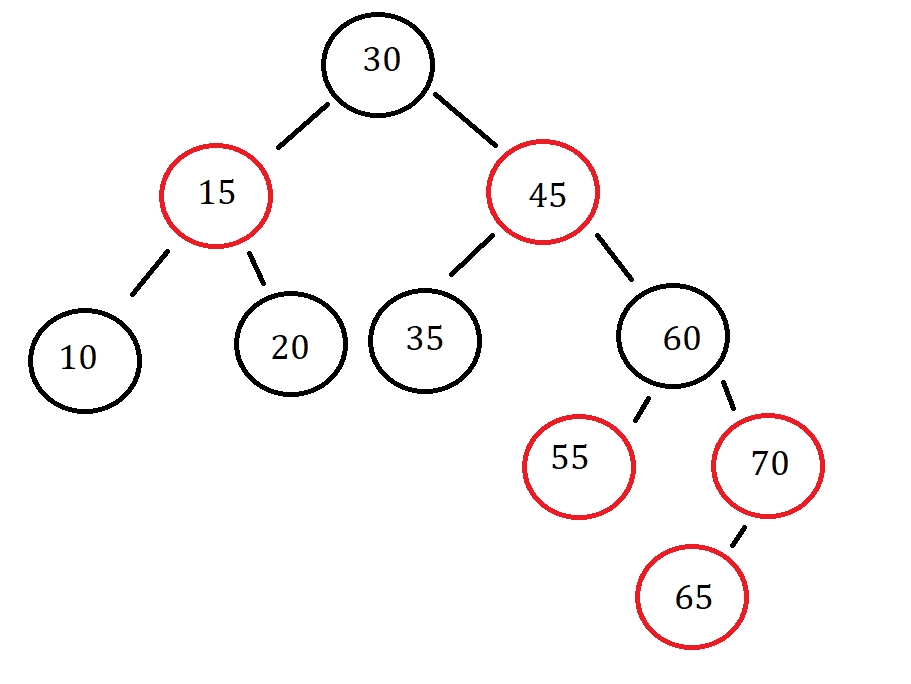
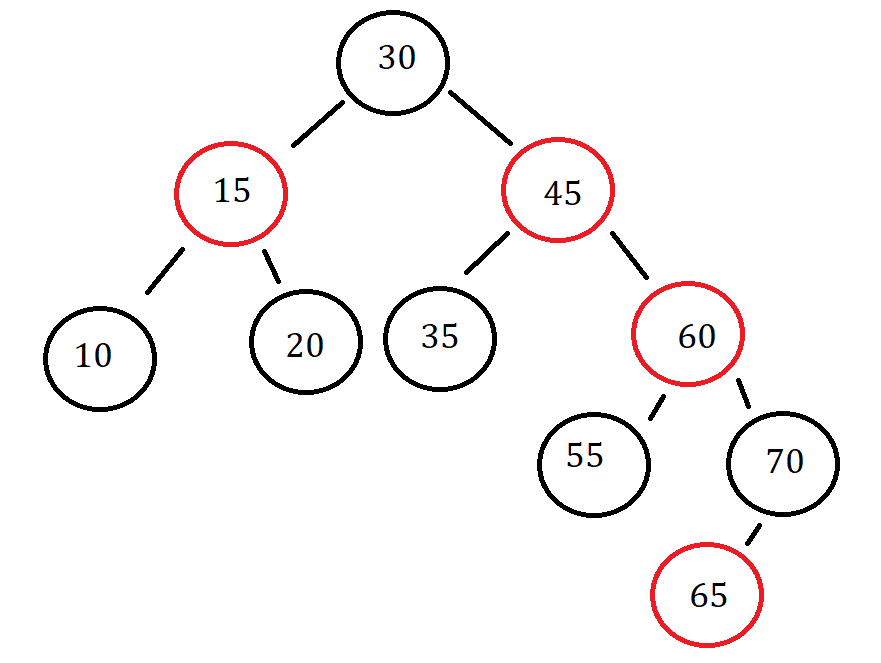
ג. הסבר: מכניסים את 50 אדום ומקבלים אבא ובן אדומים. מכיוון שהדוד שחור (null) וה 50 הוא בן שמאלי לאבא 55 שהוא שמאלי ל 60 ולכן זה מצב של LL ומכאן נעשה רוטציה ימינה סביב 60, נצבע את 60 באדום ואת 55 בשחור לפי האלגוריתם ונקבל עץ תקין.

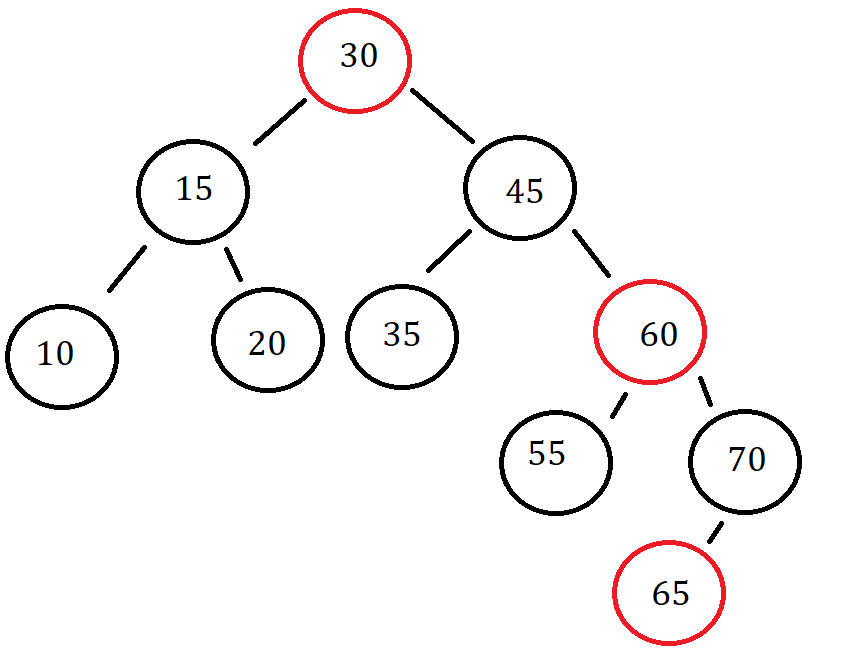
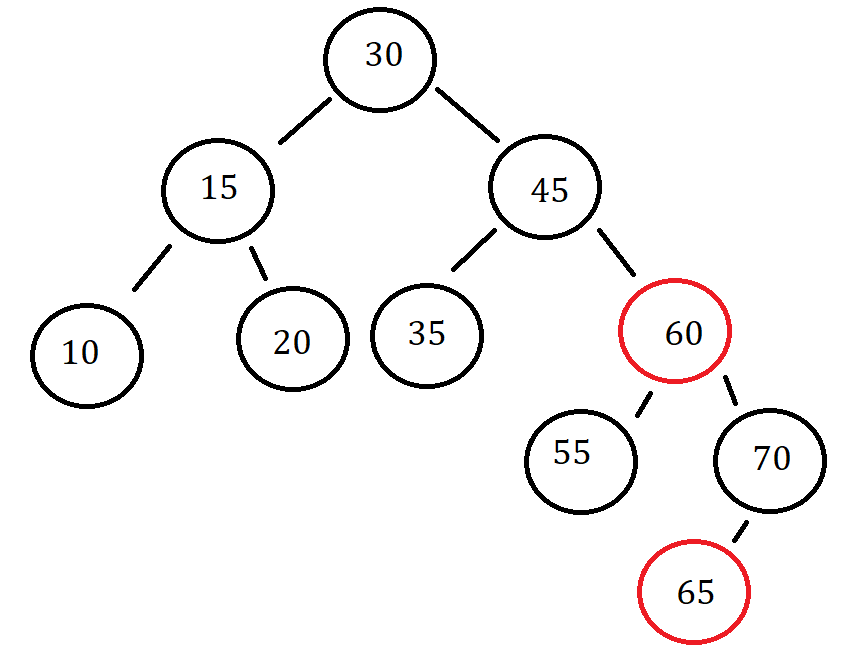


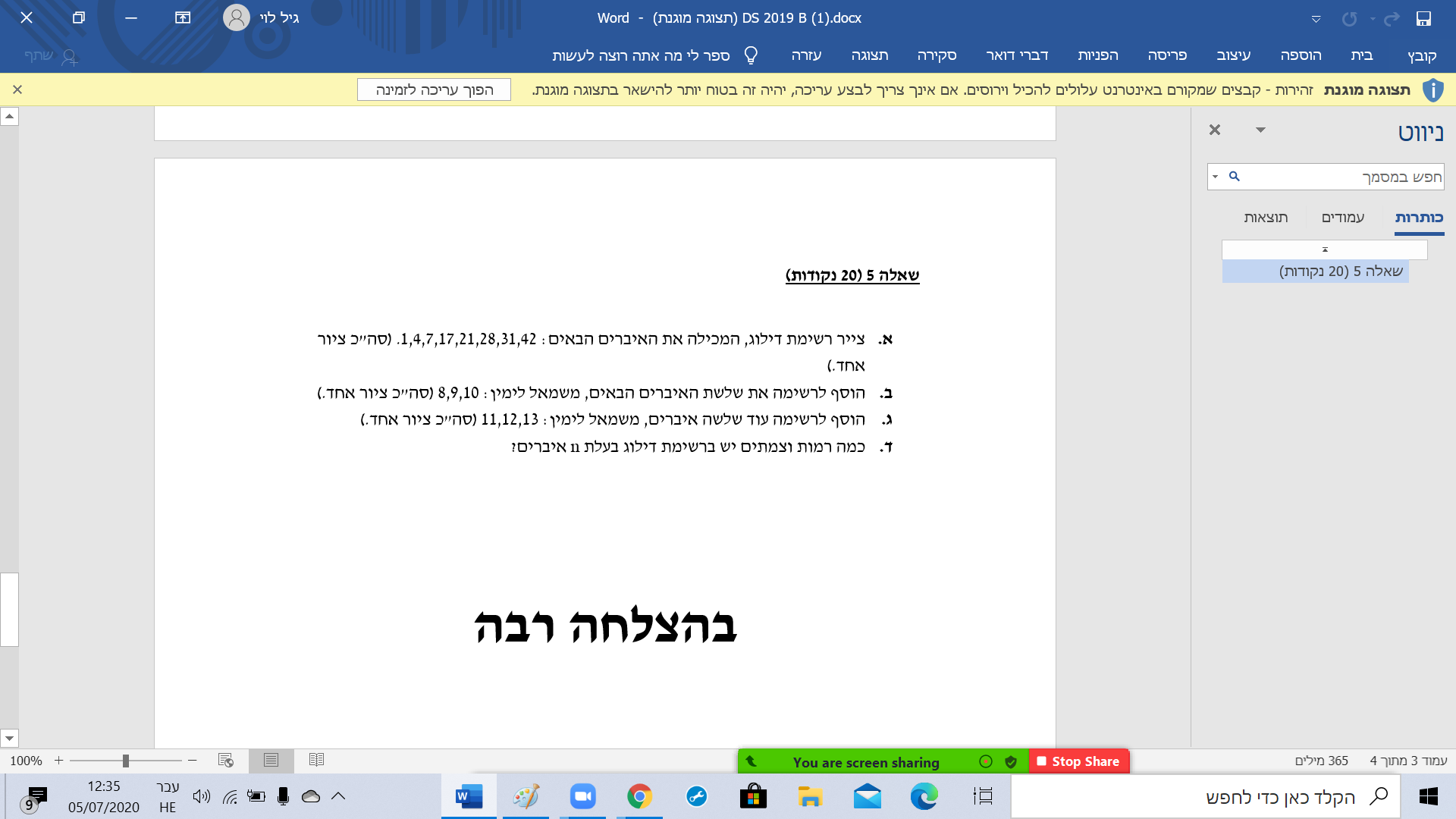


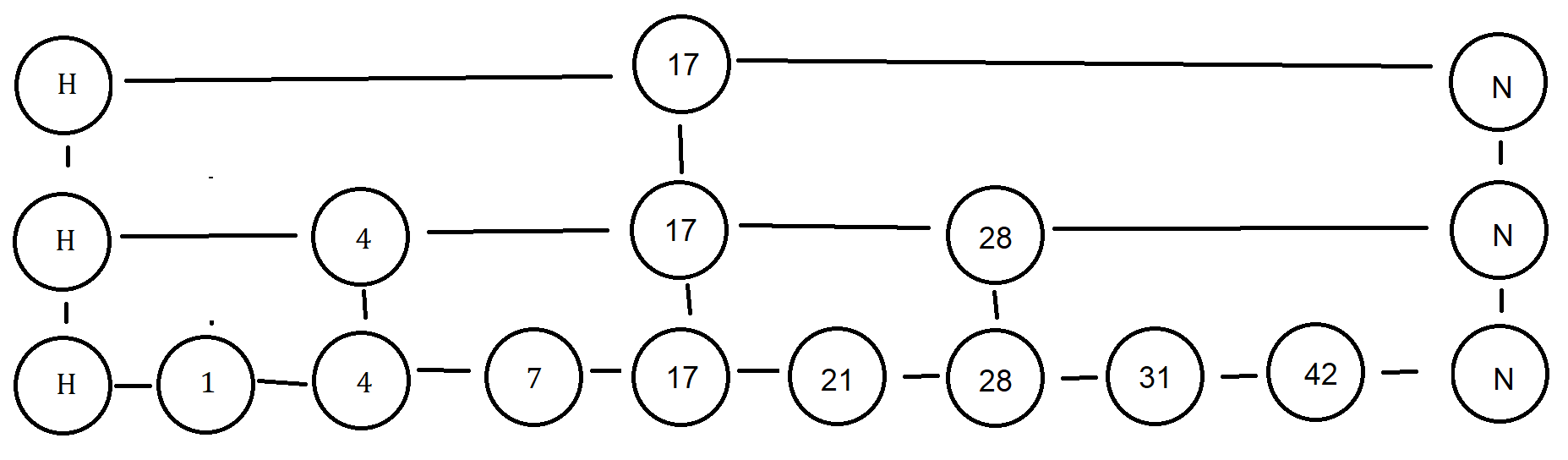
ד. נכניס את 65 אדום מתחת ל 70 (משמאל) לפי הכנסה לעץ חיפוש. קיבלנו אבא ובן אדומים. הדוד אדום ולכן נבצע החלפת צבעים: הדוד 55 והאבא 70 ייצבעו בשחור והסבא 60 יצבע באדום. לאחר מכן, לפי האלגוריתם, ממשיכים לתקן מהסבא. 60 הוא אדום והאבא 45 גם אדום. מכיוון שהדוד 15 הוא אדום שוב נחליף צבעים: האבא 45 והדוד 15 ייצבעו בשחור והסבא 30 יצבע באדום וממשיכים לתקן את הסבא 30.

מכיוון ש 30 הוא השורש, צובעים אותו בשחור ומסיימים.







פתרון:

א.

ב. ….

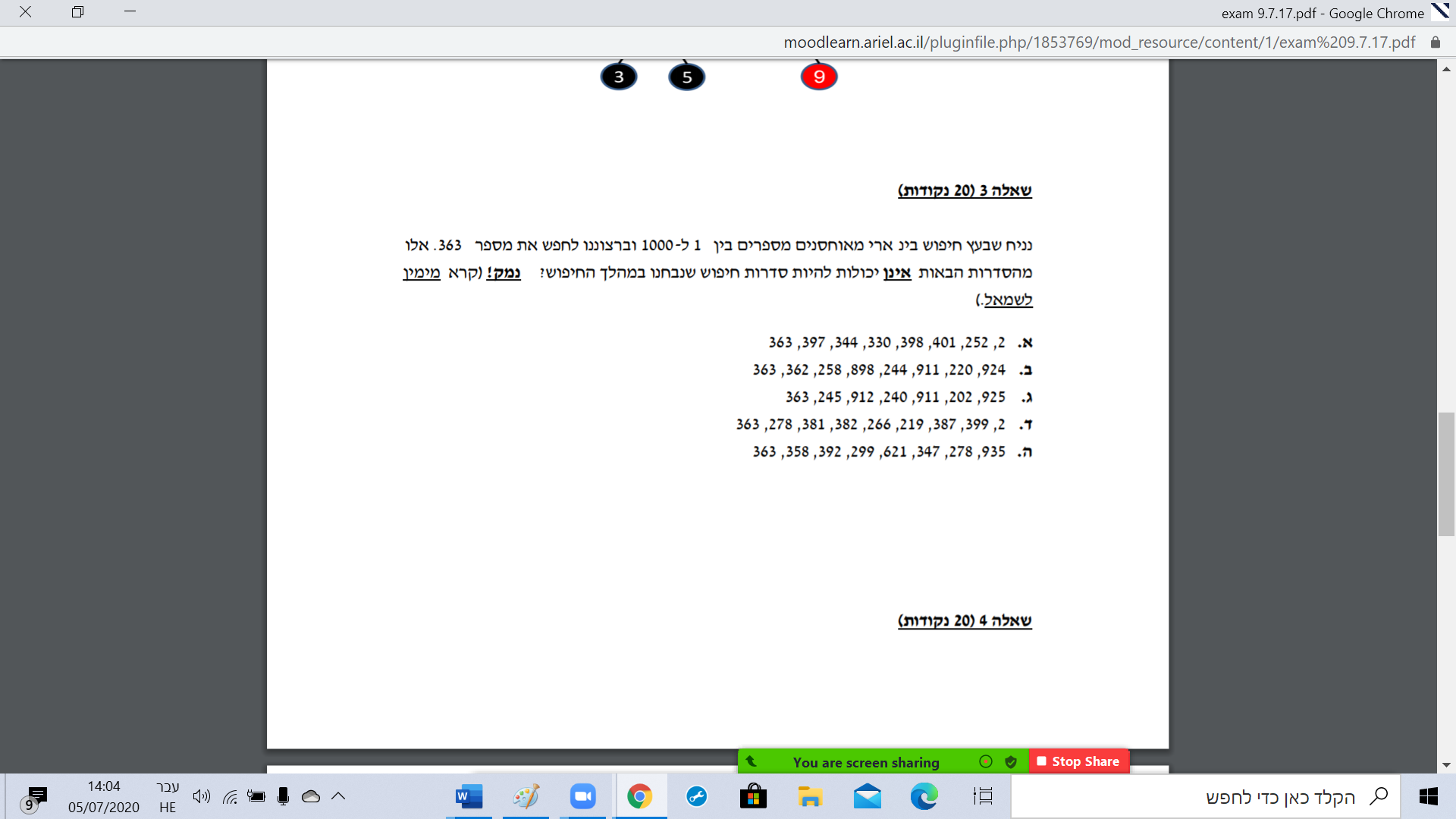
ג. ….

ד. נשים לב שברמה התחתונה יש צמתים (אחד לכל איבר)  
ברמה לפניה יש לכל הפחות צמתים ולכל היותר . לפניה יש לפחות ולכל היותר .  
מכאן, ברמה ה יש לכל הפחות ולכל היותר .  
לכל מספר הרמות יהיה לכל היותר: ומכאן: ומכאן: .

מספר הצמתים יהיה: לכל הפחות

ולכל היותר:

מבחן 2017 ב מועד א



א. 2 הוא השורש, 363 גדול מ 2 ולכן עברנו ימינה.

363 גדול מ 252 ולכן עברנו ימינה (כל המספרים כעת צריכים להיות גדולים מ 252)

363 קטן מ 401 ולכן עברנו שמאלה. (כל המספרים הם בין 252 ל 401)

363 קטן מ 398 ולכן עברנו שמאלה. (כל המספרים הם בין 252 ל 398)

363 גדול מ 330 ולכן עברנו ימינה (כל המספרים הם בין 330 ל 398)

363 גדול מ 344 ולכן עברנו ימינה (כל המספרים הם בין 344 ל 398)

363 קטן מ 397 ולכן עברנו שמאלה (כל המספרים הם בין 344 ל 397)

מצאנו את 363.

ב. 924 הוא השורש, הולכים שמאלה (חסם עליון 924)

220, הולכים ימינה (בין 220 ל 924)

911, הולכים שמאלה (בין 220 ל 911)

244, הולכים ימינה (בין 244 ל 911)

898, הולכים שאמלה (בין 244 ל 898)

258, הולכים ימינה (בין 258 ל 898)

362, הולכים ימינה (בין 362 ל 898)

מצאנו את 363

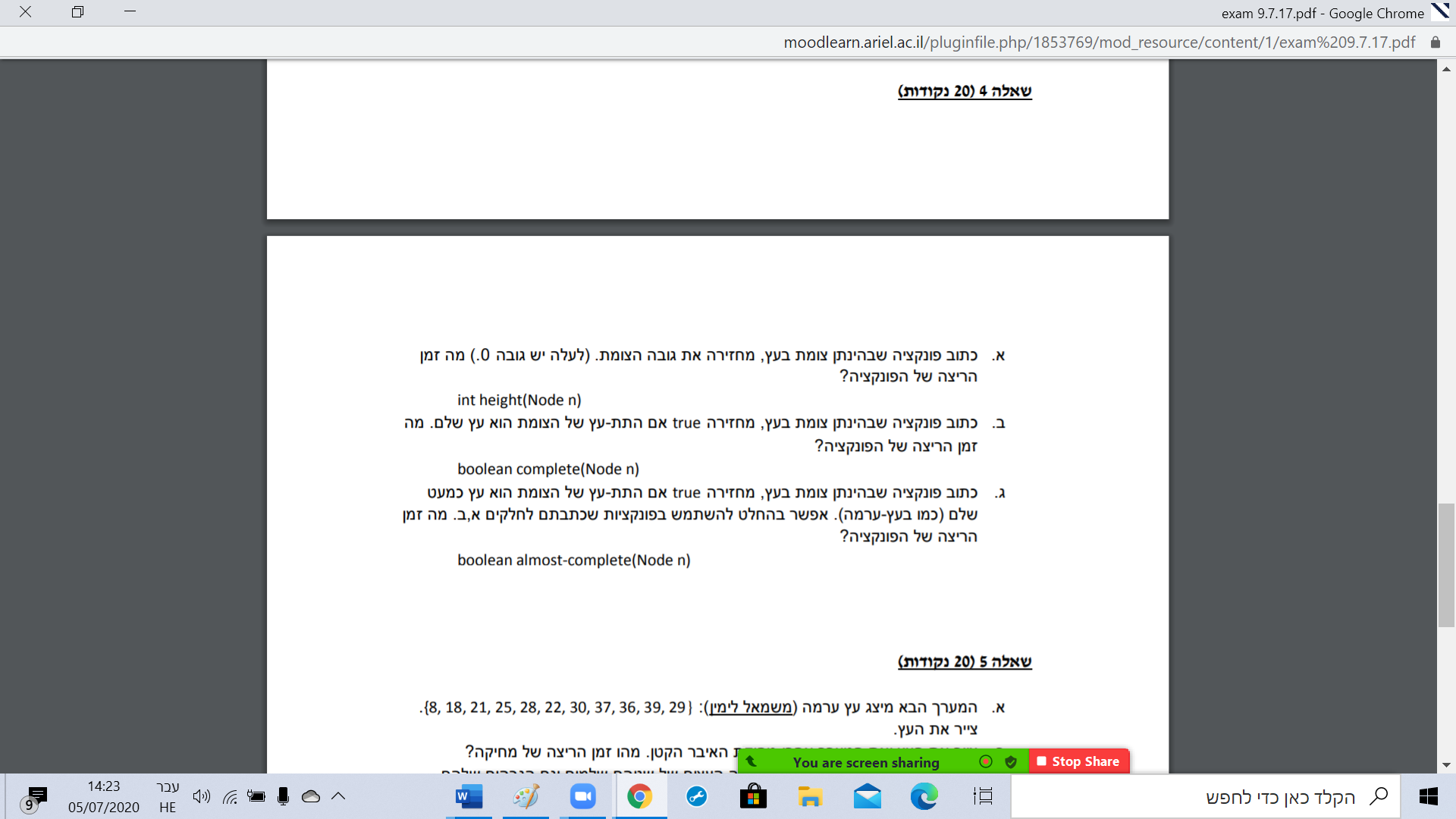
ג.925 הוא השורש, הולכים שמאלה (חסם עליון 925)

202, הולכים ימינה (בין 202 ל 925)

911, הולכים שמאלה (בין 202 ל 911)

240, הולכים ימינה (בין 240 ל 911)

912, לא הגיוני כי הוא לא בין 240 ל 911. ולכן זאת לא סדרה תקינה.



פתרון:

א.

int height(Node n)

if(n == null) return -1

else return 1 + Max(height(n.left), height(n.right))

זמן ריצה: נשים לב שכל קודקוד בתת עץ שתחת ה Node שקיבלנו יטופל פעם אחת בדיוק לכן הסיבוכיות תהיה כמספר הצמתים שיש בעץ: .

ב.

boolean complete(Node n)

int h = height(n)

return complete(n, h)

boolean complete(Node n, int h)

if(n == null) return h == -1

return complete(n.left, h-1) && complete(n.right, h-1)

זמן ריצה: קריאה לפונקציה ב-א': ואז נשים לב שכל קודקוד בתת עץ שתחת ה Node שקיבלנו יטופל פעם אחת בדיוק לכן הסיבוכיות תהיה כמספר הצמתים שיש בעץ: .  
סה"כ: .

ג.

boolean almost-complete(Node n)

Queue<Node> q = new Queue<Node>()

q.enqueue(n)

while(!q,isEmpty())

Node m = q.dequeue()

if(m == null)

while(!q,isEmpty())

if(q.dequeue() != null) return false

return true

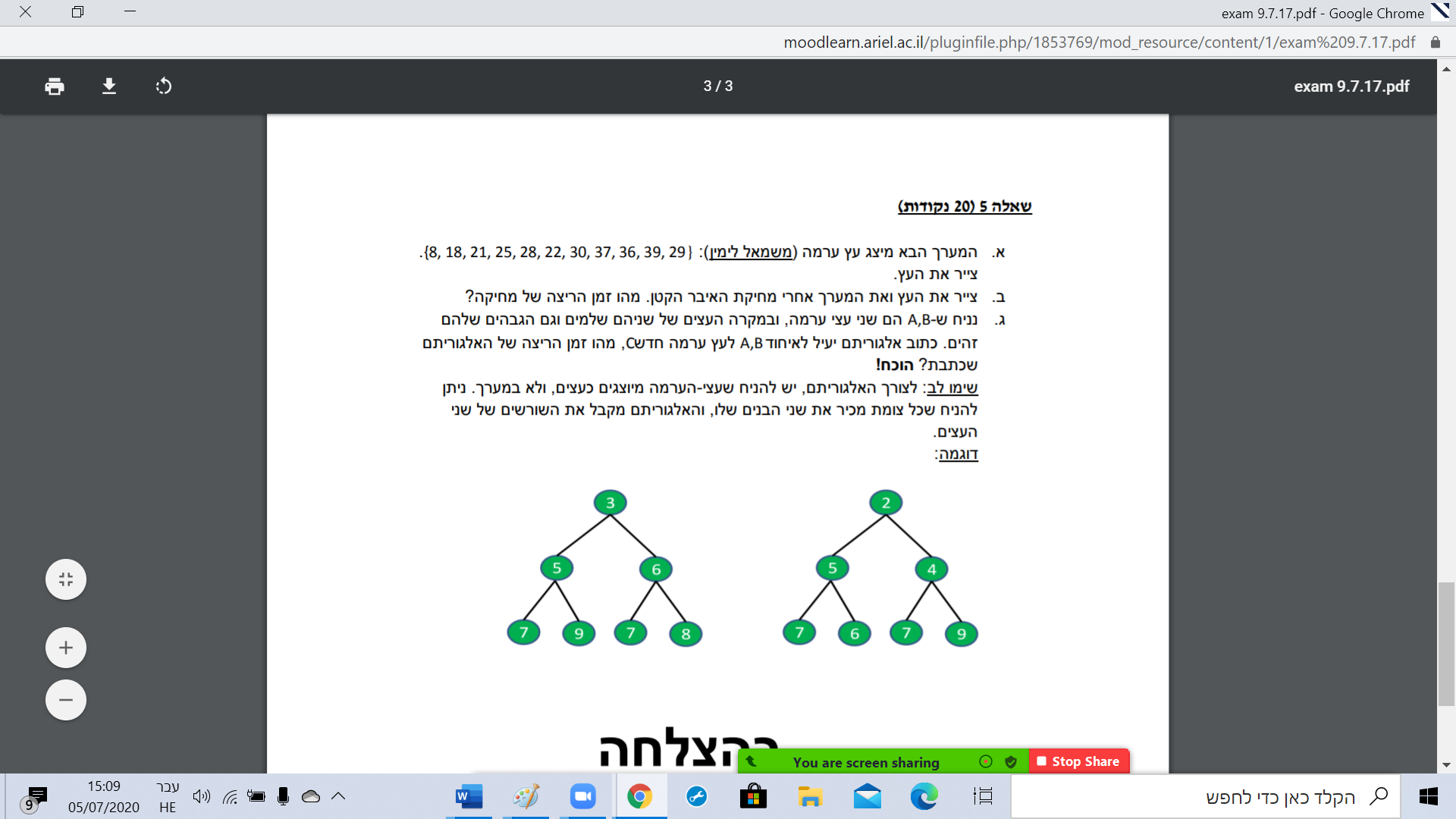
else

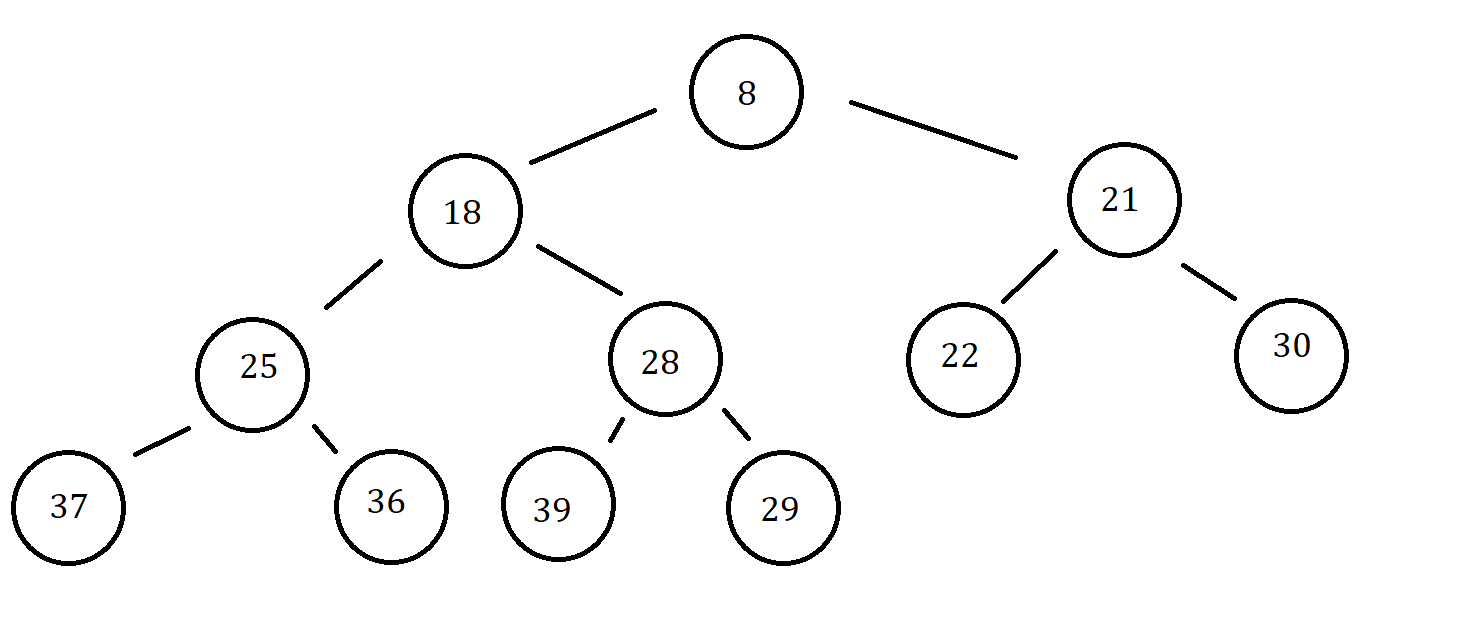
q.enqueue(m.left)

q.enqueue(m.right)

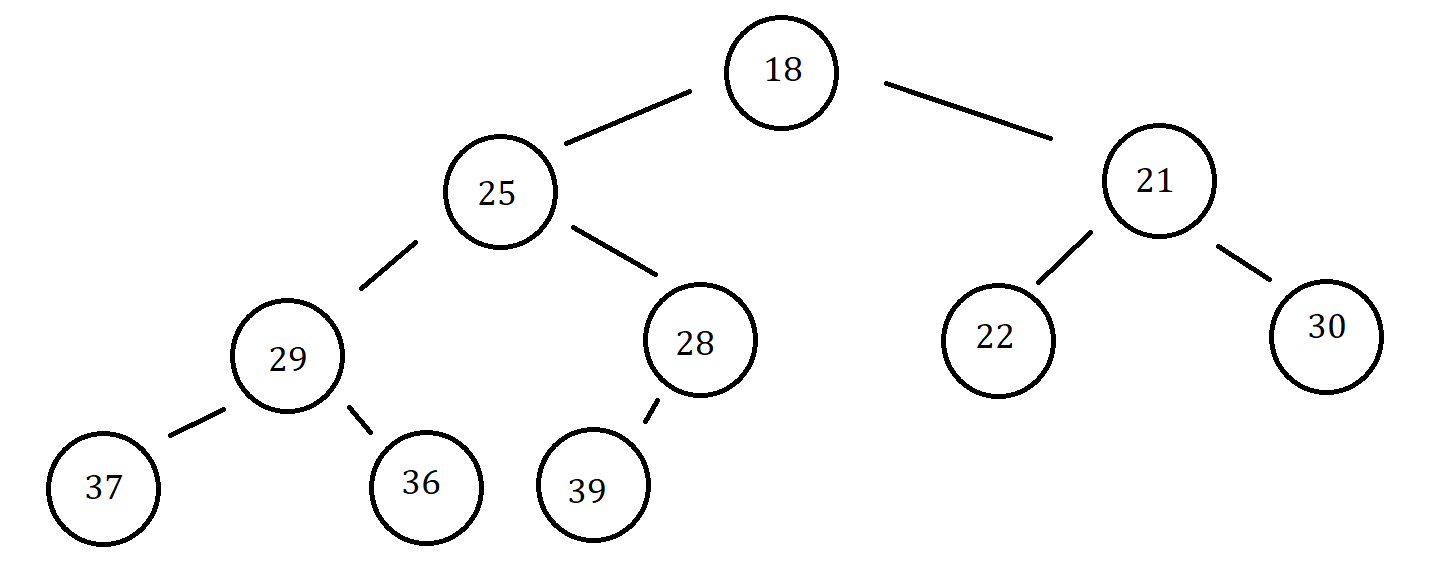
return true

זמן ריצה, כל קודקוד ייכנס ויצא מהתור פעם אחת ולכן הסיבוכיות היא: .





א.



ב.

סיבוכיות מחיקה: העברת האיבר האחרון במערך לראש העץ ודחיפתו למטה במסלול אחד - מכיוון שהעץ מאוזן, אורך המסלול הוא ולכן הסיבוכיות היא: .

ג. הרעיון: ניקח את הקודקוד הימני התחתון ביותר מאחד העצים ונהפוך אותו לשורש שהבן הימני שלו זה העץ שממנו לקחנו והבן השמאלי הוא העץ האחר. לאחר מכן נתקן ע"י הורדת האיבר מהשורש למקומו.

mergeHeaps(Node n1, Node n2)

if(n1.left == null && n2.left == null)

if(n1.data > n2.data) n2.left = n1

else n1.left = n2

else

Node m = n2

while(m.right.right != null) m = m.right

Node r = m.right

m.right = null

r.left = n1

r.right = n2

hepifyDown(r)

return r

hepifyDown(Node r)

if((r.left == null || r.data < r.left.data) && (r.right == null || r.data < r.right.data)) return

else

Node min = r.left

if(r.right != null && r.right.data < min.data) min = r.right

int temp = r.data

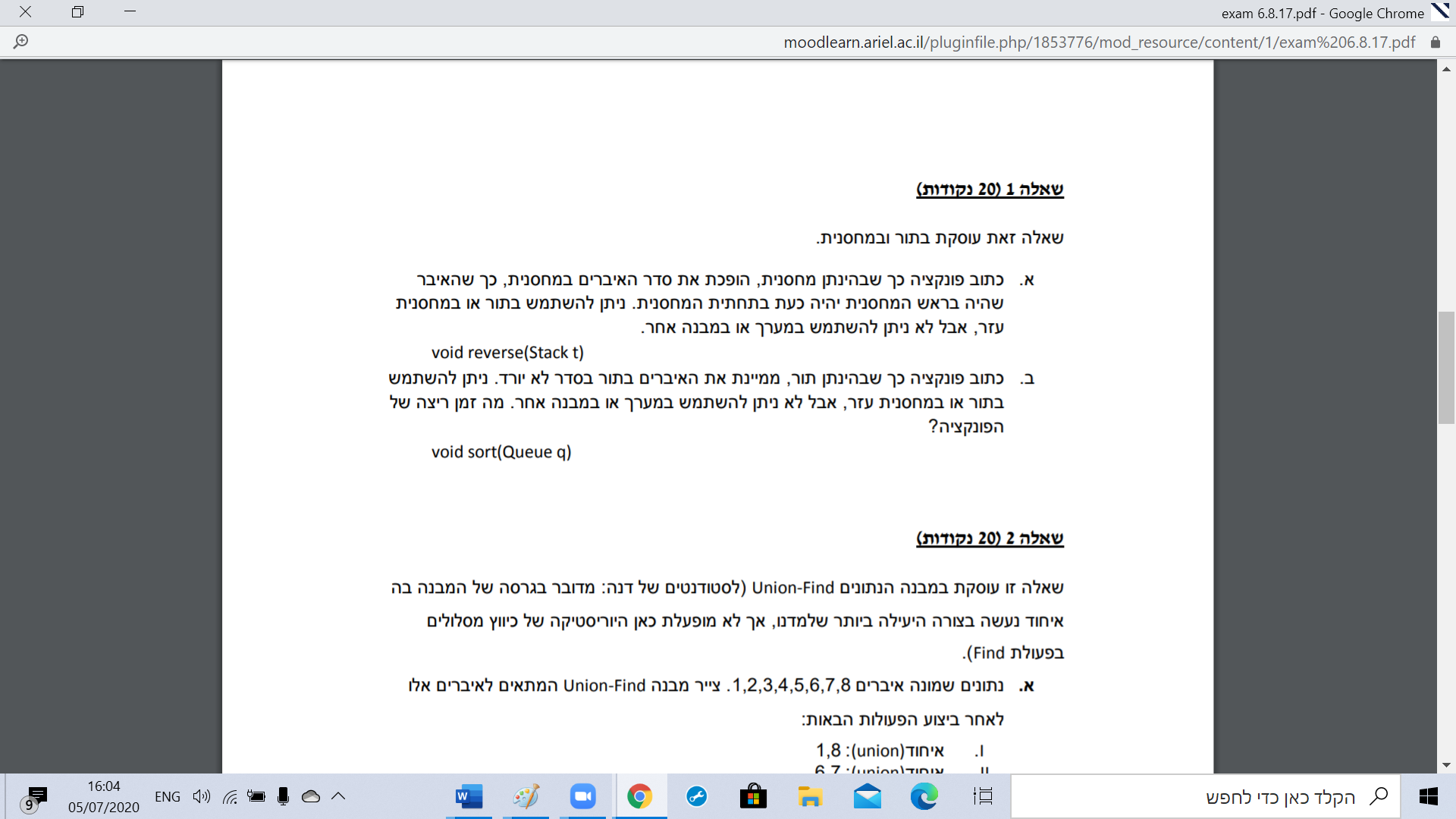
r.data = min.data

min.data = temp

hepifyDown(min)

סיבוכיות זמן ריצה: עבור המעבר הכי ימינה בעץ הראשון והעדכון של hepifyDown. אבל ולכן זה: .

מבחן 2017 ב' מועד ב'



פתרון:

א.

void reverse(Stack t)

Queue q = new Queue()

while(!t.isEmpty()) q.enqueue(t.pop())

while(!q.isEmpty()) t.push(q.dequeue())

סיבוכיות: מעבירים פעם אחת את כל האיברים מהמחסנית לתור ואז עוד פעם אחת מהתור למחסנית.

ב.

void sort(Queue q)

for(int i=0; i<q.size(); i++)

int minIndex = findMin(q, q.size() - i)

moveMinToEnd(q, minIndex)

int findMin(Queue q, int Index)

int ans = -1

int min = inf

for(int i=0; i<q.size(); i++)

int x = q.dequeue()

if(x <= min && i < Index)

ans = i

min = x

q.enqueue(x)

return ans

moveMinToEnd(Queue q, int minIndex)

int min

for(int i=0; i<q.size(); i++)

int x = q.dequeue()

if(i == minIndex)

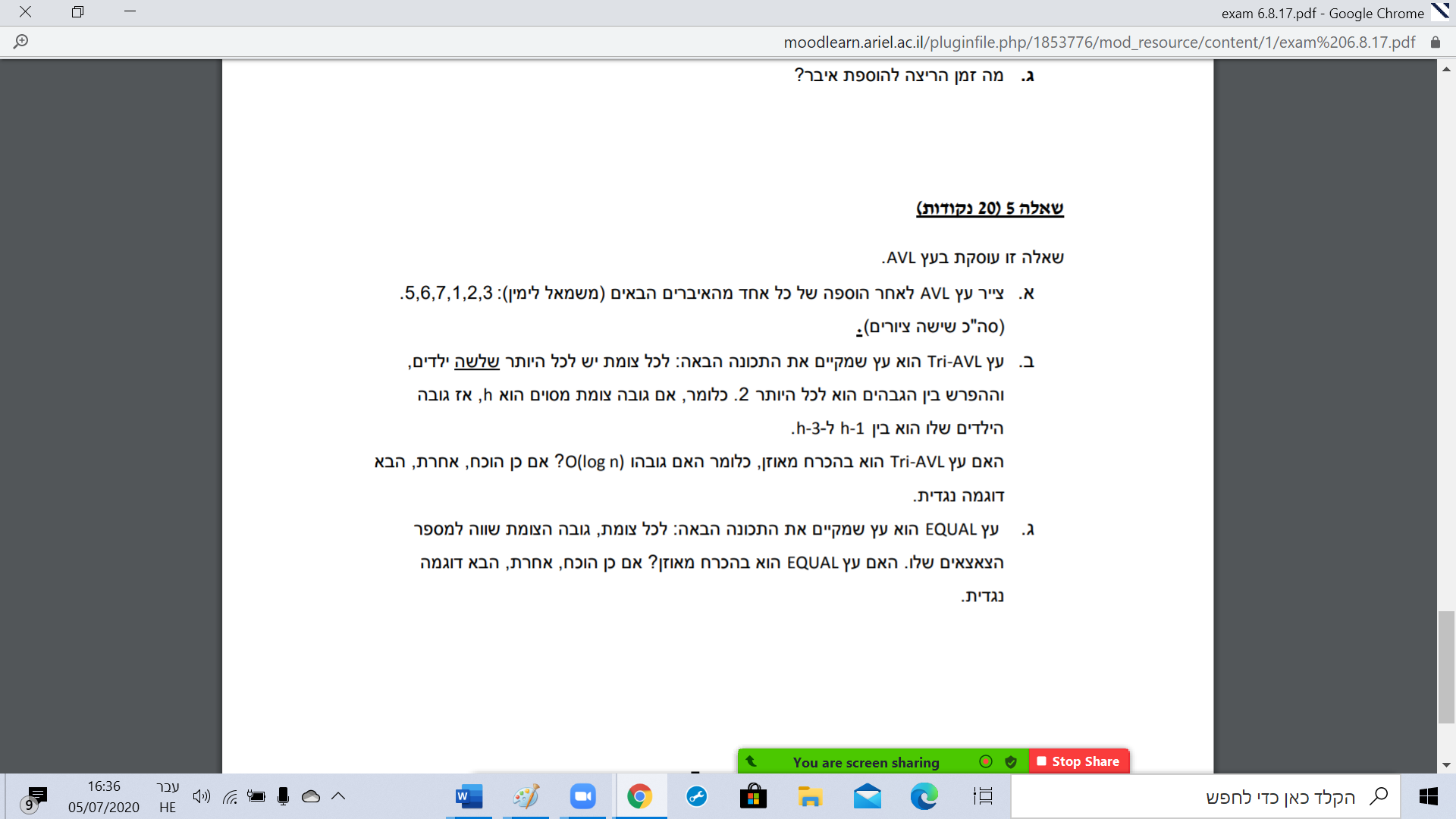
min = x

else

q.enqueue(x)

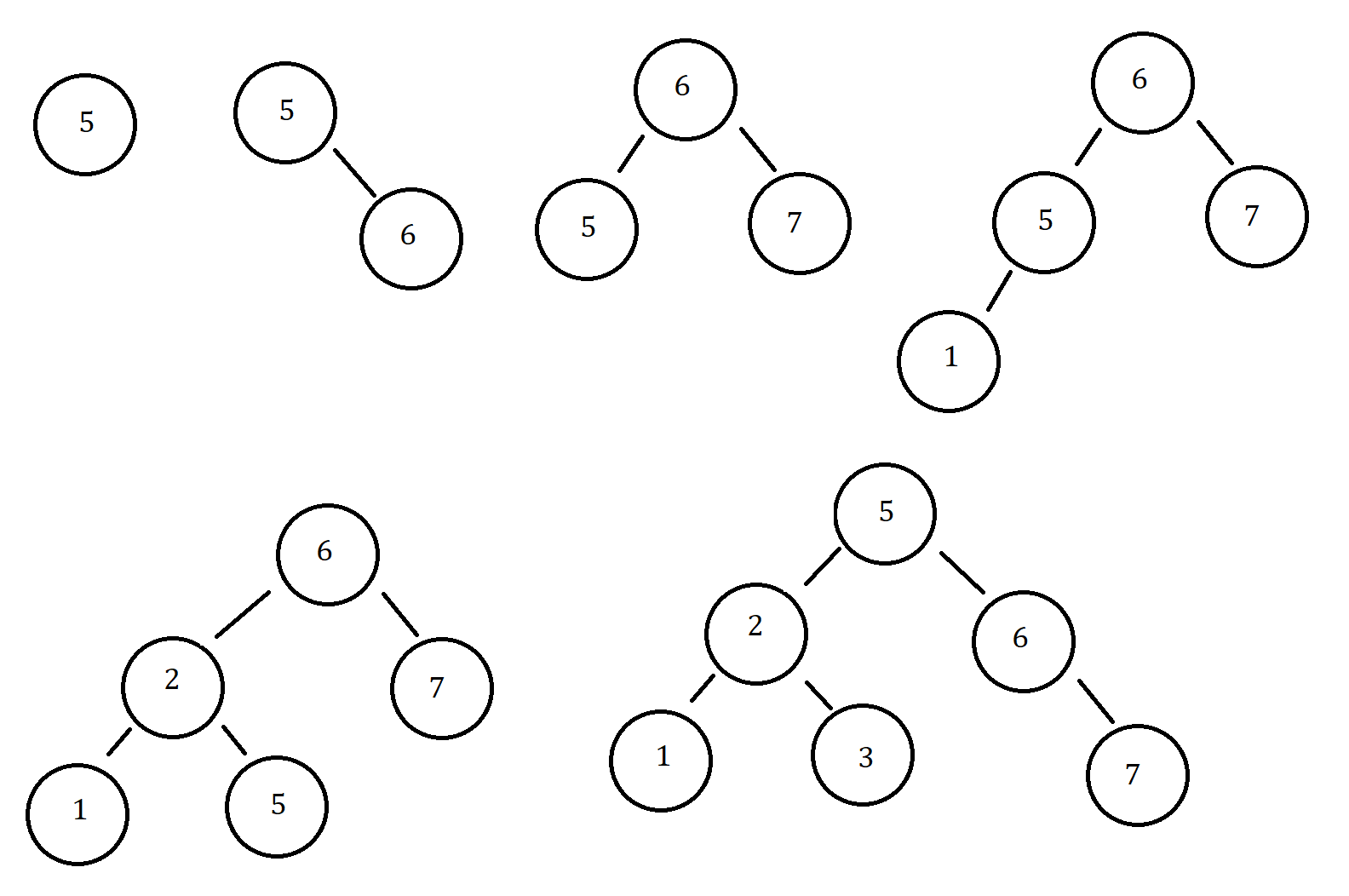
q.enqueue(min)

זמן ריצה: רצים n צעדים ובכל שלב מחפשים את המינימאלי ב ואז שמים אותו בסוף התור ולכן סה"כ נבצע: פעולות.



פתרון:

א.

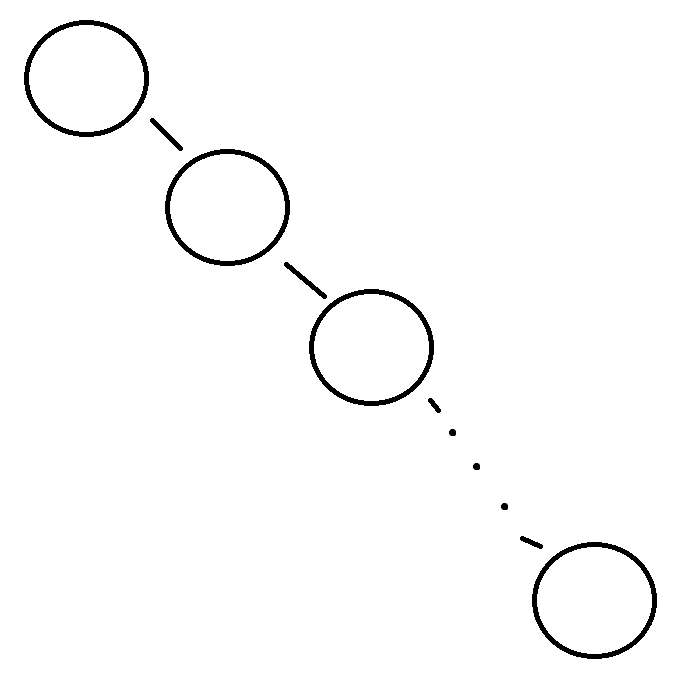


ב. נכון. הוכחה:

נסמן ב את מספר הקודקודים המינימאלי בעץ כזה בגובה h. נשים לב שבעץ בגובה h כנ"ל בעל מספר קודקודים מינימאלי. לכל קודקוד יש 3 תתי עצים בגבהים: h-1, h-3, h-3. כאשר כל אחד מתתי העצים הנ"ל הוא גם עץ תקין לפי ההגדרה (כי התנאי הוא על כל קודקוד בנפרד). לכן מתקיים: . ומכאן: . נשים לב ש:

מכאן: אם נדרוש ש: נקבל: ומכאן: . ומכאן: .

ולכן: .



ג. לא נכון.