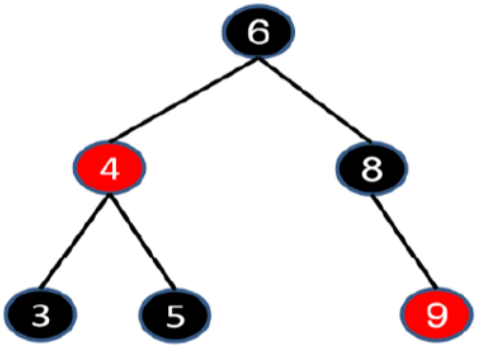
שאלות

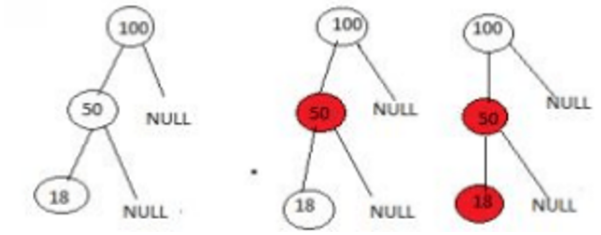
שאלה 1

נתון עץ אדום שחור , הוסף לעץ את האיברים הבאים (משמאל לימין) : $10, 7, 2, 1$. סה"כ 4 ציורים. מהו זמן הריצה לכל הכנסה?



שאלה 2

להלן עצים , כל העצים אינם עצי אדום שחור תקינים. מה מהבאים הוא הסדר הנכון שיגרום לעץ ליהיות אדום שחור תקין:



א. 50 (שחור) השורש, 18 (אדום) תת עץ שמאלי, 100 (שחור) תת עץ ימני
ב. 50 (אדום) השורש, 18 (אדום) תת עץ שמאלי, 100 (אדום) תת עץ ימני
ג. 50 (שחור) השורש, 18 (שחור) תת עץ שמאלי, 100 (אדום) תת עץ ימני
ד. 50 (שחור) השורש, 18 (אדום) תת עץ שמאלי, 100 (אדום) תת עץ ימני

שאלה 3

אחרי הוספה בעץ אדום שחור מהו מספר הרוטציות המקסימילית שנצטרך לעשות על העץ?

שאלה 4

בשאלות הבאות נתחיל לממש ערימת מקסימום. פתחו קלאס חדש בשם $MaxHeap$ המקבלת גודל מוגדר(לא ניתן לשנות את הגודל לכל אורך התרגילים הבאים) . וכתבו את השדות שלו ואת הבנאי שלו .

הערה: אני מימשתי את המחלקה הזאת עם שדה נוסף בשם $int\ last$ שמגדיר היכן האיבר האחרון כרגע נמצא.

שאלה 5

ממשו את המתודה $Heapify\_up$ לערימת מקסימום.

חתימות הפונקציה : $private\ void\ Heapify\_Up(int\ pos)$

שאלה 6

ממשו את המתודה Add לערימת מקסימום.

חתימת הפונקציה : $public\ boolean\ Add(int\ data)$

שאלה 7

ממשו את המתודה $Heapify\_down$ לערימת מקסימום.

חתימת הפונקציה : $private\ void\ Heapify\_down(int\ pos)$

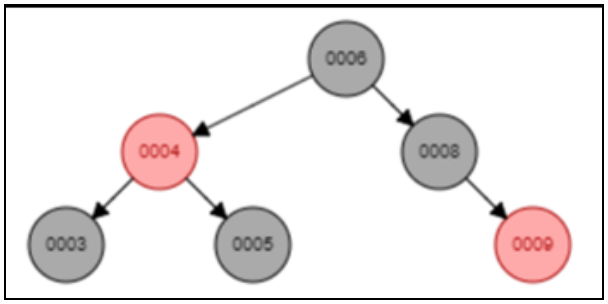
שאלה 8 , שאלת בונוס .

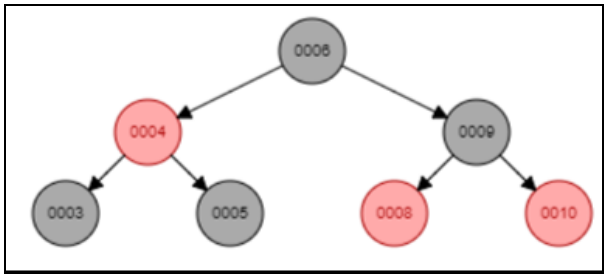
כתבו פונקציה הממזגת 2 ערימות מקסימום לערימה אחת.

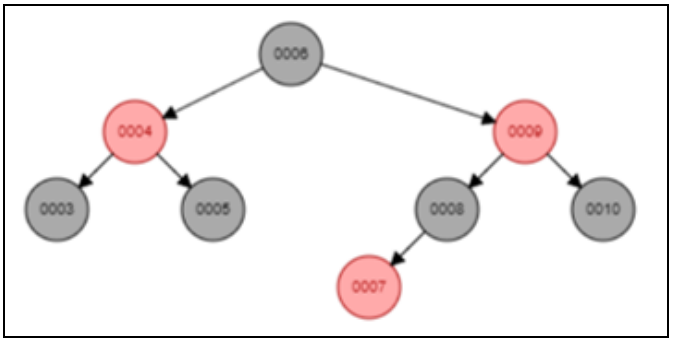
חתימת הפונקציה : $private\ static\ MaxHeap\ mergeTwoHeaps(MaxHeap\ h1,\ MaxHeap\ h2)$

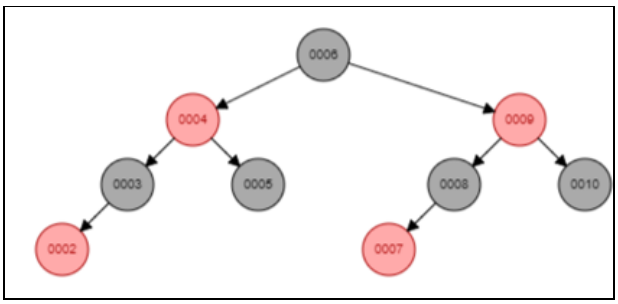
פתרונות

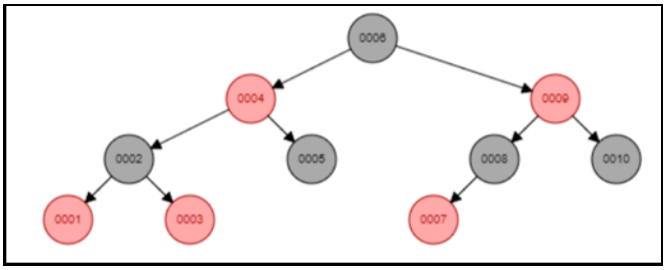
פתרון שאלה 1: זמן ההכנסה הינו $O(log(n))$ במקרה הגרוע ביותר.











פתרון שאלה 2

ד. הסבר: א וב מפרים את תנאי 3 . ג מפר את תנאי 1.

פתרון שאלה 3

מספר הרוטציות המקסימילי הוא שניים. זאת מכיוון שבמקרה הגרוע ביותר נקבל מקרה ימין שמאל או שמאל ימין וכפי שלמדנו בעזרת 2 רוטציות נתקן מקרה זה.(זכרו מקרה 1 של דוד אדום ואבא אדום שיכול להיגרר עד השורש לא גורר שום רוטציה!)

פתרון שאלה 4

```
public class MaxHeap {
  int[] arr;
  int size;

  public MaxHeap(int size) {
    this.size = size;
    arr = new int[size];
  }
}
```

פתרון שאלה 5

```
  private void Heapify_Up(int pos) {
    if (pos == 0)
      return;
    int parent = (int) (Math.floor((pos - 1) / 2));
    if (arr[pos] > arr[parent]) {
      swap(arr, pos, parent);
      Heapify_Up(parent);
    }
  }
```

פתרון שאלה 6

```
  public boolean Add(int data) {
    if (last == size - 1)
      return false;
    arr[++last] = data;
    Heapify_Up(last);
    return true;
  }
```

פתרון שאלה 7

```
1    private void swap(int[] arr, int from, int to) {
2      int tmp = arr[from];
3      arr[from] = arr[to];
4      arr[to] = tmp;
5    }
6   private void Heapify_down(int pos) {
7      int left, right;
8      left = 2 * pos + 1;
9      right = 2 * pos + 2;
10     if ((left == last) && ((arr[pos] < arr[left]))) {
11       swap(arr, pos, left);
12       return;
13     }
14     if ((right == last) && (arr[right] > arr[left]) && (arr[pos] <
   arr[right])) {
15       swap(arr, pos, right);
16       return;
17     } else if ((right == last) && (arr[right] < arr[left]) &&
   (arr[pos] < arr[left])) {
18       swap(arr, pos, left);
19       return;
20     }
21     if (left >= last || right >= last) {
22       return;
23     }
24     if ((arr[left] > arr[right]) && (arr[pos] <= arr[left])) {
25       swap(arr, pos, left);
26       Heapify_down(left);
27     } else if (arr[pos] < arr[right]) {
28       swap(arr, pos, right);
29       Heapify_down(right);
30     }
31   }
```

פתרון שאלה 8

```
1    private static MaxHeap mergeTwoHeaps(MaxHeap h1, MaxHeap h2) {
2      MaxHeap h=new MaxHeap(h1.size+h2.size);
3      h.arr = new int[h1.size+h2.size];
4      int counter =0;
5      for (int i = 0; i <= h1.last; i++) {
6        h.arr[i]=h1.arr[i];
7        counter++;
8        h.last++;
9      }
10     for (int i = 0; i <= h2.last ; i++) {
11       h.arr[counter++] = h2.arr[i];
12       h.last++;
13     }
14     //build max heap inorder to make the merged array into a merged
   heap
15     for (int i = (h.last-1)/2; i >=0 ; i--) {
16       h.Heapify_down(i);
17     }
18     return h;
19   }
```