ביה"ס למדעי המחשב ומתמטיקה האקדמית נתניה
סמסטר ב' תשע"ח

18.4.2018

מבני נתונים
פתרון תרגיל מס' 5

.1

הגדרה: נגדיר רמה של צומת בעץ בינארי T, כמספר הצמתים במסלול המחבר בין הצומת לשורש העץ. לדוגמה, הרמה של שורש העץ היא 1, והרמה של הצומת שהמפתח שלו הוא 32 בציור שבעמוד הבא היא 3.

כתוב/כתבי פסאודו-קוד של פונקציה בשם P1, יעילה ככל האפשר, אשר מקבלת כפרמטר עץ בינארי T ומדפיסה עבור כל צומת בעץ את המפתח של הצומת ואת הרמה שלו, ולבסוף בשורה האחרונה התכנית מדפיסה את סכום הרמות של כל הצמתים בעץ, עם הודעה מתאימה (כפי שמתואר בדוגמה שבהמשך).

אין חשיבות לסדר הדפסת הצמתים בפלט.

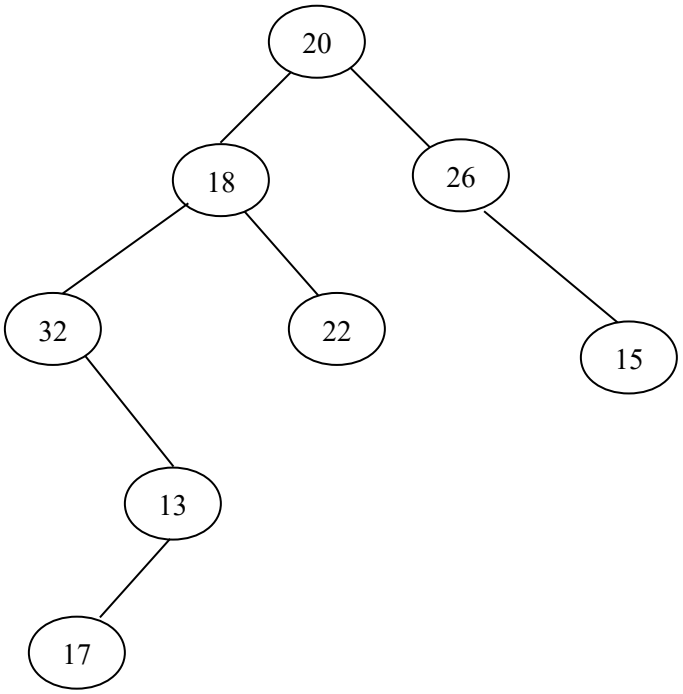
נתח/י את סיבוכיות זמן הריצה של הפונקציה שכתבת כתלות במספר הצמתים בעץ n.

הנחות ודרישות:

- אין להשתמש במבני עזר נוספים.
- מותר להשתמש במספר קבוע של משתנים (כמו למשל x,y,z).
- מותר להשתמש בפונקציות עזר, אך יש לכתוב את הפסיאודו-קוד של פונקציות העזר.
- כל צומת x בעץ מכיל את השדות הרגילים של עץ בינארי כפי שהוגדר בכיתה.

דוגמה:

יהי T עץ בינארי שמתואר בציור הבא :



לאחר הקריאה לפונקציה P1(T) יתקבל הפלט:

```
20 1
18 2
32 3
13 4
17 5
22 3
26 2
15 3
The sum of levels of all tree nodes is: 23
```

הסבר לשורה האחרונה בפלט: סכום הרמות של כל הצמתים הוא 23,
כי הוא מתקבל על ידי החישוב:

1+2+3+4+5+3+2+3=23

פתרון שאלה 1

```
P1(T)
-----
x=root(T)
if x==NULL {
    print "The sum of levels of all tree nodes is: 0"
    return
}
z=P1.1(T,0)
print "The sum of levels of all tree nodes is: z"

P1.1(T,s)
---------
x=root(T)
if x==null return 0
y1=P1.1(Tleft(x),s+1)
y2=P1.1(Tright(x),s+1)
y=y1+y2+s+1
print key(x) s+1
return y
```

ובפורמט טקסט:

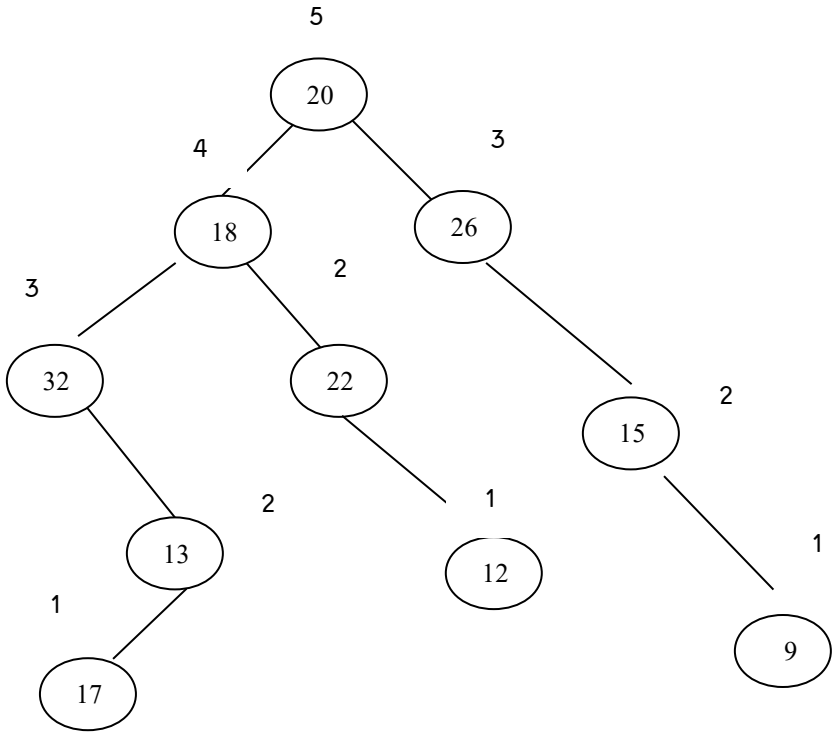
```
P1(T)
-----
x=root(T)
if x==NULL {
    print "The sum of levels of all tree nodes is: 0"
    return
}
z=P1.1(T,0)
print "The sum of levels of all tree nodes is: z"

P1.1(T,s)
---------
x=root(T)
if x==null return 0
y1=P1.1(T_left(x),s+1)
y2=P1.1(T_right(x),s+1)
y=y1+y2+s+1
print key(x) s+1
return y
```

2.

נגדיר "עץ בינארי עם גבהים" כעץ בינארי T שבו לכל צומת x בנוסף לשדות הרגילים יש שדה h(x) המכיל את גובה תת העץ ששורשו x.

לדוגמה: יהי T העץ הבא שבו לכל צומת מצוין מפתח הצומת ומעליו מספר המציין את גובה הצומת.



לכל צומת x נסמן ב- balance(x) את גורם האיזון של הצומת שמוגדר כגובה של תת העץ השמאלי של x פחות הגובה של תת העץ הימני של x. לדוגמה גורם האיזון של הצומת שהמפתח שלו 18 הוא 1, גורם האיזון של הצומת שהמפתח שלו הוא 26 הוא 2- וכן הלאה.

נגדיר שצומת x בעץ הוא צומת "טוב" אם מתקיימים שני התנאים הבאים:

1. במסלול מהאבא של הצומת x לשורש העץ יש לפחות צומת אחד בעל גורם איזון חיובי.

2. בתת העץ ששורשו x (תת העץ כולל גם את x) יש לפחות שני צמתים בעלי גורם איזון שלילי.

כתוב/כתבי פסאודו-קוד של פונקציה בשם P2, <u>יעילה ככל האפשר</u>, אשר מקבלת כפרמטר עץ בינארי T ומדפיסה את כל הצמתים הטובים בעץ, (אין חשיבות לסדר הדפסת הצמתים).

נתח/י את סיבוכיות זמן הריצה של הפונקציה שכתבת כתלות במספר הצמתים בעץ n.

<u>הנחות ודרישות:</u>

- אין להשתמש במבני עזר נוספים.
- מותר להשתמש במספר קבוע של משתנים (כמו למשל x,y,z).
- מותר להשתמש בפונקציות עזר, אך יש לכתוב את הפסיאודו-קוד של פונקציות העזר.
- כל צומת x בעץ מכיל את השדות הרגילים של עץ בינארי כפי שהוגדר בכיתה ובנוסף את השדה h(x) שמכיל את הגובה של תת העץ ששורשו x.

<u>דוגמה:</u>

יהי T העץ הבינארי שמתואר בציור שבדף הקודם.

לאחר הקריאה לפונקציה P2(T) יתקבל הפלט:

18 26

<u>הסבר לפלט</u>: 18 הוא צומת טוב כי במסלול מהאבא שלו לשורש העץ יש צומת אחד 20 עם גורם איזון חיובי, ולכן התנאי הראשון מתקיים, ובתת העץ ששורשו 18 יש שני צמתים עם גורם איזון שלילי 22 ו- 32 ולכן גם תנאי 2 מתקיים.
26 הוא צומת טוב כי במסלול מהאבא שלו לשורש העץ יש צומת אחד 20 עם גורם איזון חיובי, ובתת העץ ששורשו 26 יש שני צמתים עם גורם איזון שלילי 26 ו- 15.

פתרון שאלה 2

```
P2(T)
-----
x=root(T)
if x==NULL return
P2.1(T,false)


P2.1(T,f)
---------
x=root(T)
if x==null {y.h=0; y.c=0; return y}
f1=f
if h(left(x))>h(right(x)) f1=true
y1=P1.1(Tleft(x),f1)
y2=P1.1(Tright(x),f1)
y=y1+y2
if h(left(x))<h(right(x)) y++
if (f==true && y □ 2) print key(x)
return y
```

ובפורמט טקסט:

```
P2(T)
-----
x=root(T)
if x==NULL return
P2.1(T,false)


P2.1(T,f)
---------
x=root(T)
if x==null {y.h=0; y.c=0; return y}
f1=f
if h(left(x))>h(right(x)) f1=true
y1=P1.1(T_left(x),f1)
y2=P1.1(T_right(x),f1)
y=y1+y2
if h(left(x))<h(right(x)) y++
if (f==true && y ≤ 2) print key(x)
return y
```

3.

עבור שני צמתים x ו- y בעץ T נגדיר את הסכום של החיבור בינהם כסכום המפתחות של הצמתים במסלול בעץ T שמחבר בין x ל- y.
(שימו לב שבכל עץ T יש רק מסלול אחד שמחבר בין שני צמתים).

כתוב/כתבי פסאודו-קוד של פונקציה בשם p3, יעילה ככל האפשר, אשר מקבלת כפרמטר עץ בינארי T שמכיל לפחות שני צמתים, ומדפיסה את המספר הגדול ביותר ששוה לסכום החיבור של שני צמתים שונים כלשהם בעץ.

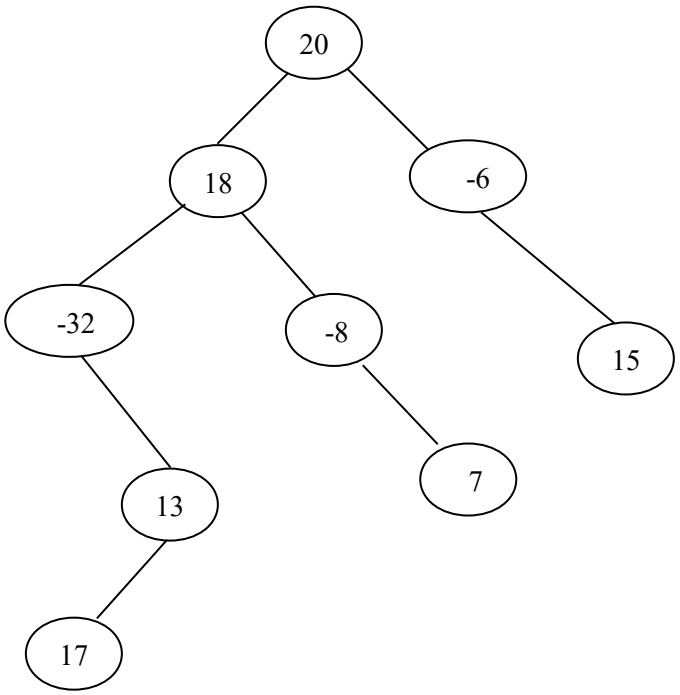
נתח/י את סיבוכיות זמן הריצה של הפונקציה שכתבת כתלות במספר הצמתים בעץ n.

הנחות ודרישות:

- אין להשתמש במבני עזר נוספים.
- מותר להשתמש במספר קבוע של משתנים (כמו למשל x,y,z).
- מותר להשתמש בפונקציות עזר, אך יש לכתוב את הפסיאודו-קוד של פונקציות העזר.
- כל צומת x בעץ מכיל את השדות הרגילים של עץ בינארי כפי שהוגדר בכיתה.

דוגמה:

יהי **T** עץ בינארי שמתואר בציור הבא :



לאחר הקריאה לפונקציה **P3(T)** יתקבל הפלט:

47

הסבר לפלט: המספר **47** שווה לסכום מפתחות הצמתים במסלול שמחבר בין **18** ל- **15** לפי החישוב הבא:

18+20-6+15=47

מאחר וזהו הסכום הגדול ביותר שמחבר בין שני צמתים שונים בעץ, זוהי התשובה שמתקבלת.

פתרון שאלה 3

הפתרון של השאלה מורכב יחסית משתי סיבות:

1. הדרישה ששני הצמתים במסלול יהיו שונים זה מזה, ולכן עבור צומת שהוא עלה אין בעצם תשובה אפשרית ולכן אין לבצע קריאה רקורסיבית על עץ שהוא עלה. לצורך כך לפני הקריאה הרקורסיבית בודקים אם העץ שקוראים אליו הוא עלה ובמידה וכן לא מבצעים את הקריאה הרקורסיבית.

2. אין דרישה בשאלה שהצמתים יהיו חיוביים, ולכן מסלול קצר יותר יכול להיות עדיף על מסלול ארוך יותר שמכיל צמתים שליליים. לצורך כך החישוב של root_to_node הוא של המסלול הגדול ביותר מהשורש לאיזשהו צומת בתת העץ של השורש ולא בהכרח לעלה בתת עץ זה.

להלן הפתרון:

```
P3(T)
------
x==root(T)
if (x==NULL || (left(x)==NULL && right(x)==NULL) {
    print "The tree does not have at least 2 nodes"
    return
}
z=P3.1(T)
print z.max
```

```
P3.1(T)
-------
x=root(T)
if (left(x)==NULL && right(x)==NULL) {
  y.max=key(x) /* this value will not be used
  y.root_to_node=key(x) /* this value will not be used
  return y
}
```

```
if (left(x)!=NULL && right(x)==NULL) {
  if is_leaf(left(x))==true {
     y.max=key(x)+key(left(x))
     y.root_to_node=max{key(x),key(x)+key(left(x))}
     return y
  }
  y1=P3.1(Tleft(x))  /* here left(x) is not a leaf
  y.max=max { y1.max, key(x)+y1.root_to_node}
  y.root_to_node=max{key(x),key(x)+y1.root_to_node}
  return y
}

if (left(x)==NULL && right(x)!=NULL) {
  if is_leaf(right(x))==true {
     y.max=key(x)+key(right(x))
     y.root_to_node=max{key(x),key(x)+key(right(x))}
     return y
  }
  y2=P3.1(Tright(x))  /* here right(x) is not a leaf
  y.max=max { y2.max, key(x)+y2.root_to_node}
  y.root_to_node=max{key(x),key(x)+y2.root_to_node}
  return y
}

/* here left(x)!=NULL && right(x)!=NULL
y1=P3.1(Tleft(x))
y2=P3.1(Tright(x))

if (is_leaf(left(x))==true && is_leaf(right(x))==true){
  y.max=key(x)+max{key(left(x)),key(right(x)}
  y.root_to_node=
     max{key(x),key(x)+key(left(x)),key(x)+key(right(x))}
  return y
}
if (is_leaf(left(x))!=true && is_leaf(right(x))==true){
  y.max=max{y1.max, key(x)+key(right(x)),
            key(x)+y1.root_to_node,
            key(x)+y1.root_to_node+key(right(x))}
  y.root_to_node=max{key(x),key(x)+key(right(x)),
                     key(x)+y1.root_to_node}
  return y
}
```

```
if (is_leaf(left(x))==true && is_leaf(right(x))!=true){
  y.max=max{y1.max, key(x)+key(left(x)),
            key(x)+y2.root_to_node,
            key(x)+y2.root_to_node+key(left(x))}
  y.root_to_node=max{key(x),key(x)+key(left(x)),
                     key(x)+y2.root_to_node}
  return y
}
/* here is_leaf(left(x))!=true && is_leaf(right(x))!=true)
y.max=max{y1.max,y2.max,
          key(x)+y1.root_to_node,
          key(x)+y2.root_to_node,
          key(x)+y1.root_to_node+y2.root_to_node}
y.root_to_node=max{key(x),key(x)+y1.root_to_node,
                   key(x)+y2.root_to_node}
return y


is_leaf(x)
----------
if (left(x)==NULL && right(x)==NULL) return true
return false
```

להלן התכנית בפורמט טקסט:

```
P3(T)
------
x==root(T)
if (x==NULL || (left(x)==NULL && right(x)==NULL) {
    print "The tree does not have at least 2 nodes"
      return
}
z=P3.1(T)
print z.max

P3.1(T)
-------
x=root(T)
if (left(x)==NULL && right(x)==NULL) {
  y.max=key(x) /* this value will not be used
  y.root_to_node=key(x) /* this value will not be used
  return y
}
```

```
if (left(x)!=NULL && right(x)==NULL) {
  if is_leaf(left(x))==true {
     y.max=key(x)+key(left(x))
       y.root_to_node=max{key(x),key(x)+key(left(x))}
       return y
  }
  y1=P3.1(Tleft(x))  /* here left(x) is not a leaf
  y.max=max { y1.max, key(x)+y1.root_to_node}
  y.root_to_node=max{key(x),key(x)+y1.root_to_node}
  return y
}
if (left(x)==NULL && right(x)!=NULL) {
  if is_leaf(right(x))==true {
     y.max=key(x)+key(right(x))
       y.root_to_node=max{key(x),key(x)+key(right(x))}
       return y
  }
  y2=P3.1(Tright(x))  /* here right(x) is not a leaf
  y.max=max { y2.max, key(x)+y2.root_to_node}
  y.root_to_node=max{key(x),key(x)+y2.root_to_node}
  return y
}
/* here left(x)!=NULL && right(x)!=NULL
y1=P3.1(Tleft(x))
y2=P3.1(Tright(x))

if (is_leaf(left(x))==true && is_leaf(right(x))==true){
  y.max=key(x)+max{key(left(x)),key(right(x)}
  y.root_to_node=
     max{key(x),key(x)+key(left(x)),key(x)+key(right(x))}
  return y
}
if (is_leaf(left(x))!=true && is_leaf(right(x))==true){
  y.max=max{y1.max, key(x)+key(right(x)),
            key(x)+y1.root_to_node,
                 key(x)+y1.root_to_node+key(right(x))}
  y.root_to_node=max{key(x),key(x)+key(right(x)),
                     key(x)+y1.root_to_node}
  return y
}
if (is_leaf(left(x))==true && is_leaf(right(x))!=true){
  y.max=max{y1.max, key(x)+key(left(x)),
            key(x)+y2.root_to_node,
                 key(x)+y2.root_to_node+key(left(x))}
  y.root_to_node=max{key(x),key(x)+key(left(x)),
                     key(x)+y2.root_to_node}
  return y
}
```

```
/* here is_leaf(left(x))!=true && is_leaf(right(x))!=true)

y.max=max{y1.max,y2.max,
          key(x)+y1.root_to_node,
          key(x)+y2.root_to_node,
          key(x)+y1.root_to_node+y2.root_to_node}
y.root_to_node=max{key(x),key(x)+y1.root_to_node,
                   key(x)+y2.root_to_node}
return y

is_leaf(x)
----------
if (left(x)==NULL && right(x)==NULL) return true
return false
```

4. שאלה זו הופיע במועד א 2017

<u>הגדרה</u>: נגדיר עץ בינארי אדום לבן, כעץ בינארי שבו לכל צומת x, בנוסף לשדות הרגילים, יש שדה צבע color(x) שהוא אדום (red) או לבן (white).

נזכיר שעבור צומת x בעץ בינארי T אנחנו מסמנים ב- T_x את תת העץ של T ששורשו x, (תת העץ T_x כולל גם את הצומת x).

נגדיר שצומת x בעץ בינארי אדום לבן הוא צומת <u>טוב</u> אם סכום המפתחות של כל הצמתים האדומים בתת העץ T_x, גדול יותר מסכום המפתחות של כל הצמתים הלבנים בתת העץ T_x.

כתוב/כתבי פסאודו-קוד של תכנית בשם P1, <u>יעילה ככל האפשר</u>, אשר מקבלת כפרמטר עץ בינארי אדום לבן T, ומדפיסה תחילה את כל הצמתים הטובים בעץ (אין חשיבות לסדר הדפסת הצמתים), ולאחר מכן התכנית מדפיסה את מספר הצמתים הטובים בעץ.

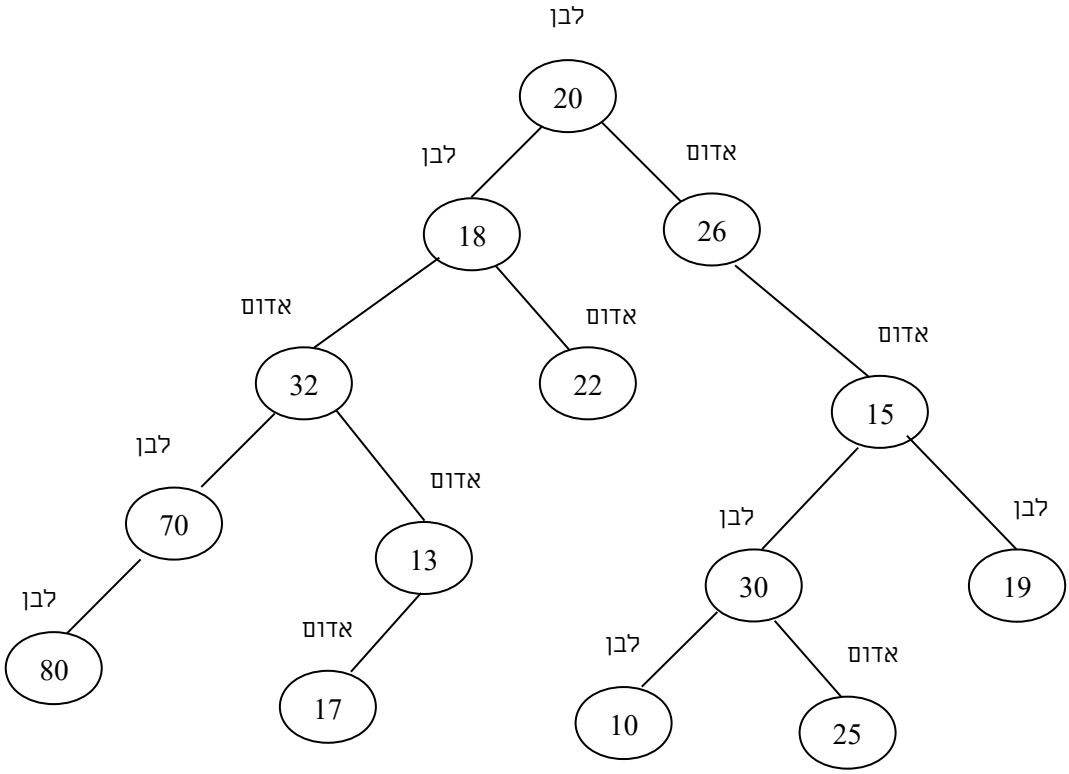
נתח/י את סיבוכיות זמן הריצה של הפונקציה שכתבת כתלות במספר הצמתים בעץ n.

<u>הנחות ודרישות:</u>

1. אין להשתמש במבני עזר נוספים.
2. מותר להשתמש במספר קבוע של משתנים (כמו למשל x,y,z).
3. מותר להשתמש בפונקציות עזר, אך יש לכתוב את הפסיאודו-קוד של פונקציות העזר.
4. כל צומת x בעץ מכיל את השדות הרגילים של עץ בינארי כפי שהוגדר בכיתה ובנוסף שדה color(x) שמציין את צבע הצומת x.

דוגמה:

יהי T עץ בינארי אדום לבן שמתואר בציור הבא :



לאחר הקריאה לפונקציה P1(T) יתקבל הפלט:

```
17 13 22 26 25
5
```

<u>הסבר לפלט</u>: צמתים 17,13,22,25 הם טובים כי אם אדומים ואין מתחתם צמתים לבנים. צומת 26 הוא טוב כי סכום הצמתים האדומים בתת העץ T_{26} הוא: $26+15+25=66$ **וסכום זה גדול יותר מסכום הצמתים הלבנים בתת העץ T_{26} שהוא:** $30+10+19=59$.

שאר הצמתים בעץ אינם טובים, למשל צומת 20 אינו טוב כי סכום הצמתים האדומים בתת העץ T_{20} הוא: $32+13+17+22+26+15+25=150$

והוא קטן יותר מסכום הצמתים הלבנים בתת העץ T_{20} שהוא: $80+70+18+20+30+10+19=247$.

מאחר ויש בעץ 5 צמתים טובים מודפס בסוף המספר 5 שמצין את מספר הצמתים הטובים בעץ.

פתרון שאלה 4

```
P4(T)
-----
y=P4.1(T)
print y.good

P4.1(T)
-------
x=root(T)
if x==NULL {y.good=0; y.red=0; y.white=0; return}
y1=P4.1(Tteft(x))
y2=P4.1(Tright(x))
y.red=y1.red+y2.red
y.white=y1.white+y2.white
if color(x)==red {y.red+=key(x)}
if color(x)==white {y.white+=key(x)}
y.good=y1.good+y2.good
if y.red > y.white {y.good++; print key(x)}
return y
```

ובפורמט טקסט:

```
P4(T)
-----
y=P4.1(T)
print y.good

P4.1(T)
-------
x=root(T)
if x==NULL {y.good=0; y.red=0; y.white=0; return}
y1=P4.1(Tteft(x))
y2=P4.1(Tright(x))
y.red=y1.red+y2.red
y.white=y1.white+y2.white
if color(x)==red {y.red+=key(x)}
if color(x)==white {y.white+=key(x)}
y.good=y1.good+y2.good
if y.red > y.white {y.good++; print key(x)}
return y
```