הנדסת תוכנה 094129

תרגיל בית 4

שימוש בתבניות ושגיאות.

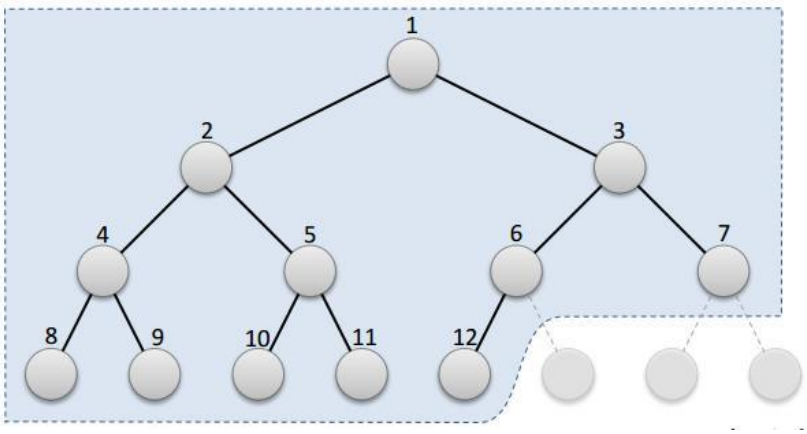
תאריך אחרון להגשה: 26/08/2019 בשעה 23:55

מתרגל אחראי: אדם גולדבראיך

נושא התרגיל: בניית עץ חיפוש בינארי

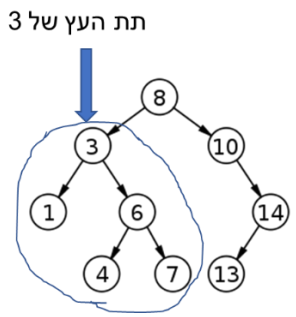
מבוא אינטואיטיבי מתורת הגרפים:

מבחינה אינטואיטיבית, עץ מושרש הוא גרף שמורכב מצומת מיוחד הנקרא "שורש" שאליו מחוברים אפס או יותר בנים. לצומת v מסוים אוסף כל בניו, נכדיו ניניו וכיוצא בזה, נקראים הצאצאים של הצומת. צומת שאין לו ילדים יקרא עלה. במקרה שבו מובטח שלכל צומת יש לכל היותר שני בנים, העץ יקרא עץ בינארי.



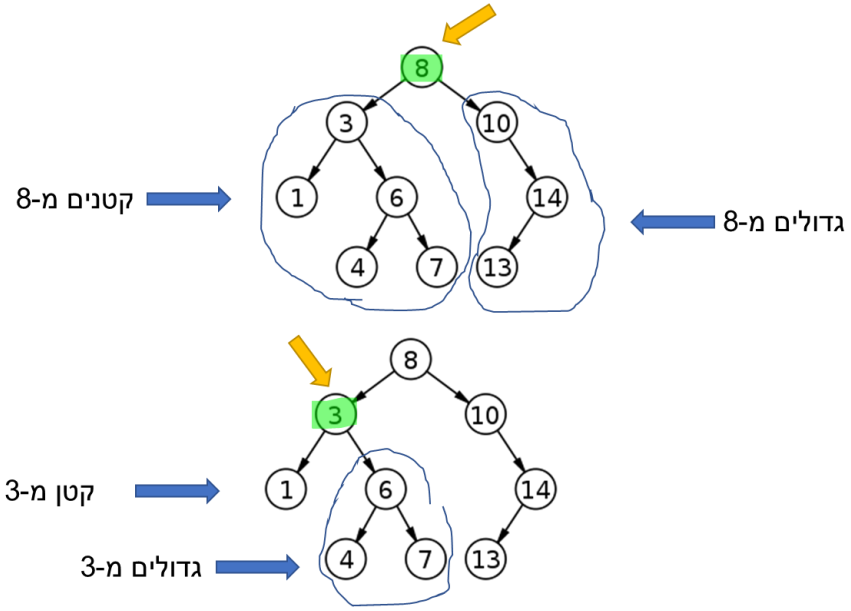
באיור זה הצומת שמסומן ב-1 הינו שורש העץ וצמתים מספר 8,9,10,11 ו- 12 הינם עלים.

תת העץ של צומת v כלשהו הוא הצומת v, כל צאצאיו ואוסף הקשתות המחברים בינהם.



כמעט עץ חיפוש בינארי

עץ חיפוש בינארי הוא עץ הנתונים הפופולרי ביותר. בעץ זה יש לכל קודקוד שני בנים לכל היותר, ימני ושמאלי. לכל צומת יש שדה שנקרא **מפתח (key)** כאשר מוגדר יחס סדר על המפתחות. הבן הימני, יחד עם כל צאצאיו, נמצא ביחס הסדר אחרי הקודקוד שממנו הוא יוצא, ואילו הבן השמאלי וכל צאצאיו קודמים לקודקוד האב ביחס הסדר. כלל זה נקרא התכונה של עצי חיפוש בינאריים.



מתודת החיפוש (Search) בעץ חיפוש בינארי משתמשת באופן ישיר בתכונה של עצי חיפוש בינאריים. האלגוריתם יעבור על כל קודקוד וקודקוד החל מהשורש, וינוע ימינה כאשר מפתח הקודקוד הנוכחי במסלול קטן מהמפתח של הקודקוד המוכנס, ושמאלה במקרה ההפוך עד אשר יתקיים שוויון.

```
Tree_Search(x, k)
if x == nil or x.key == k then
    return x
if k < x.key then
    return Tree_Search(x.left, k)
else return Tree_Search(x.right, k)
```

פסאודו-קוד 1: חיפוש בעץ חיפוש בינארי

שימו לב, הקטע הבא כתוב בפסאודו קוד, משמעותו של NIL שקול ל- null ב- C++.

גם כדי להוסיף קודקוד חדש לעץ יעבור האלגוריתם על כל קודקוד וקודקוד החל מהשורש, וינוע ימינה כאשר מפתח הקודקוד הנוכחי במסלול קטן מהמפתח של הקודקוד המוכנס, ושמאלה במקרה ההפוך. האלגוריתם ייעצר ויבצע את ההוספה במקום שבו המסלול ייגמר.

```
Tree_Insert(T, z)
if T.root == NIL then
    T.root = z
else
    y = T.root
    x = NIL
    while y ≠ NIL do
        x = y
        if z.key < y.key then
            y = y.left
        else y = y.right
    z.p = x
    if z.key < x.key then
        x.left = z
    else x.right = z
```

פסאודו-קוד 2: הכנסה של צומת חדש לעץ חיפוש בינארי

במקרה שבו נרצה למחוק **עלה**, נחפש את הצומת ע"פ המפתח, נוודא שאין לו ילדים ואז פשוט נסיר אותו מהעץ. מבנה הנתונים שלנו יקרא כמעט עץ חיפוש בינארי כיוון שהוא יתמוך במחיקת עלים, אך לא יתמוך במחיקה של צומת כללי, כיוון שפעולה זו דורשת הבנה עמוקה יותר מבחינה אלגוריתמית מהסקופ של הקורס שלנו, בסוף הסמסטר הבא, לאחר שתלמדו את הקורס מבני נתונים ואלגוריתמים תהיו מסוגלים להרחיב את המימוש של מבנה נתונים זה לעץ חיפוש בינארי רגיל בפשטות.

הפעולה החשובה האחרונה היא סיור סדור בעץ (inorder traversal). אלגוריתם הסיור יקבל תמיד את השורש ויתחיל לרוץ ממנו. האלגוריתם מכיל פונקציה רקורסיבית שנקראת visit אשר מקבלת כפרמטר אובייקט מטיפוס צומת. בשלב הראשון הפונקציה תיקרא רקורסיבית על הבן השמאלי, לאחר מכן תופס הצומת עצמה ואז הפונקציה תיקרא רקורסיבית פעם נוספת על הבן הימני.

ראו את הפסאודו-קוד הבא:

```
function visit(node)
   if node.left  != null then visit(node.left)
   print "key:", node.key, "data:", node.item ;
   if node.right != null then visit(node.right)
```

פסאודו-קוד 3: החלק הרקורסיבי בפונקציית inorderTraversal.

מבני הנתונים של התוכנה:

בתרגיל זה אתם נדרשים לממש כמעט עץ חיפוש בינארי שיודע לקבל כל זוג מ 3 סוגי הקלט: int, float, string הם יסומנו ע"י G ו- T, במידה והטיפוס הוא string נניח כי הערך נתון כמילה אחת ללא רווחים. בכל ריצה של התוכנית המחסנית תהיה מסוג אחד בלבד (ראו דוגמא בהמשך) בדומה למה שלמדנו על MAP.

במימוש המחלקות, עליכם להשתמש בטיפוסי הנתונים הבאים:

typedef Node<T, G> > TreeNode;

כחלק מהפתרון עליכם לממש את המחלקות הבאות, בהתאם לממשק המתואר להלן:

Class name	Data members
Tree	Node<T,G>*_root; size_t _size;
Node	Node * _ parent; Node* _left; Node * _right; const T _key; G _item;

Class name	Public member methods		
	Methods Name	Description	Raises
Tree	Tree();	בונה מחלקה. עץ החיפוש הבינארי מאותחל להיות ריקה	
	bool isEmpty() const;	בודק אם העץ ריק	
	size_t size() const;	מחזיר את מספר האיברים אשר מאוחסנים בעץ	

	Node<T,G>* search (const T& key) const;	מחזיר את התוכן (item) של האיבר אם נמצא. אם לא נמצא מפתח זה, תיזרק שגיאה עם ההודעה " Key was not found"	range_error
	void insert(const T& key,const G& item);	מכניס איבר שהמפתח שלו _key והתוכן שלו הוא _item. אם קיים איבר עם מפתח זהה, זורק שגיאה עם ההודעה " Key not unique ".	range_error
	void deleteLeaf(const T& key);	מוחק את העלה שהמפתח שלו הוא _key אם לא נמצא מפתח זה, תיזרק שגיאה עם ההודעה " Key was not found" אם המוצת שנמצא אינו עלה, תיזרק שגיאה עם ההודעה "Internal node was found"	range_error
	void inOrderTraversal () const;	מדפיס את המפתח והתוכן של האיברים בעץ לפי סדר המפתחות שלהם. איבר אחד בשורה שבה יודפס: "Key: X data:Y" כאשר X ו-Y הם ערכי המפתח והתוכן המתאימים. לפני ההדפסה, מדפיס את השורה: "Binary search tree content:"	
	~ Tree();	הורס המחלקה	
Node	Node(const T& key, const G& item, Node* parent=NULL);	בונה המחלקה. מאתחל את ערך האיבר.	
	const G& getItem() const;	מחזיר את התוכן (_item) של האיבר	
	const T& getKey() const;	מחזיר את המפתח של האיבר	
	Node* getLeftChild () const;	מחזיר מצביע לבן השמאלי	
	Node* getRightChild () const;	מחזיר מצביע לבן הימני	
	~Node();	הורס המחלקה	

- יש להדפיס את הפלט בדיוק כפי שניתן שמתואר בהמשך.
- יש לדאוג לשחרור זיכרון כנדרש

- **נדרש לבצע העמסת אופרטור פלט (<<) עבור הדפסת צומת.**

הקלט לתוכנה:

הסימולציה קולטת את נתוני הפקודות מערוץ הקלט הסטנדרטי (cin). הקלט מחולק לפקודות וכל פקודה תופסת שורה אחת בדיוק בקלט. סדר ביצוע הפקודות חייב להיות לפי סדר הופעתן בקלט (FIFO).

הסימולציה מתנהלת על ידיד הפקודות הבאות:

תיאור הפקודה	מבנה שורת הפקודה	פרמטרים
הגדרת עץ חיפוש בינארי חדש	new TYPE1 TYPE2	TYPE1 – סוג המפתחות. TYPE2- סוג התוכן מקבלת אחד מהערכים הבאים: int, float, string רק פקודה אחת כזאת מתקבלת בכל ריצה. במידה ו-TYPEX הוא לא אחד מהנרשמים מעלה יש לזרוק שגיאת runtime_error עם הטקסט: "Bad Key Type: "TYPE1 (יש להחליף את TYPE עם הערך שהתקבל, ראו דוגמא) או "Bad Item Type: "TYPE2 במידה והקלט הראשון איננו new יש לזרוק שגיאת runtime_error עם הטקסט: "Bad Input: "input (כאשר input מייצג את הפקודה שהתקבלה, ראו דוגמא)
דחיפת איבר למיכל	insert Key Item	הכנסת איבר חדש עם ערך Item והמפתח Key. אם Key או Item אינו תואם לטיפוס המוגדר יש לזרוק runtime_error עם הטקסט: "Bad input type!"
מחיקת עלה עם המפתח KEY	deleteLeaf Key	Key – מפתח של הצומת אותו נרצה למחוק במידה ותופסים שגיאה, יש להדפיס בפלט שגיאה את התוכן (עדיף להשתמש בנתיב הפלט cerr)
חיפוש צומת ע"פ ערך המפתח	search Key	Key – מפתח של הצומת אותו נרצה למצוא יש להדפיס את תוכן האיבר שנמצא עם התחילית "found: "

		במידה ותופסים שגיאה, יש להדפיס בפלט שגיאה את התוכן (עדיף להשתמש בנתיב הפלט cerr)
בדיקה אם העץ ריק	empty	ללא יש להדפיס, אם ריק: “The tree is empty” אחרת: “The tree is not empty”
הדפסת מספר הצמתים בעץ	size	יש להדפיס: “Tree size:” SIZE כאשר SIZE הוא הערך שמוחזר
הדפסת התוכן ע"פ הסדר שהוגדר באלגוריתם, שימו לב להגדרת הפונקציה לדרך שבה הפלט צריך להיות מיוצג.	inorderTraversal	ללא
סיום הפעולה	quit	ללא
שגיאה		אם מתקבלת פקודה אחרת יש להדפיס הודעת שגיאה (עדיף להשתמש בנתיב הפלט cerr) “Bad command”

הפלט של המערכת:

דוגמא לאינטרקציה עם התוכנה, קלט באדם, פלט בירוק ושגיאות בכחול.

```
new int string
insert 5 Bob
insert 22 Alice
insert 17 Sean
insert 8 Oded
insert 3 Shir
insert 9 Maor
insert 1 Rom
empty
size
inorderTraversal
deleteLeaf 1
deleteLeaf 9
deleteLeaf 8
deleteLeaf 22
inorderTraversal
quit
```

```
The tree is not empty
Tree size: 7
Binary search tree content:
key: 1        data:Rom
key: 3        data:Shir
key: 5        data:Bob
key: 8        data:Oded
key: 9        data:Maor
key: 17       data:Sean
key: 22       data:Alice
Binary search tree content:
key: 3        data:Shir
key: 5        data:Bob
key: 17       data:Sean
key: 22       data:Alice
```

קובץ השגיאות:

```
The node is not a leaf!
```

דרישות המימוש:

1. המימוש חייב להכיל לפחות את הקבצים הבאים:
   i. main.cpp – כפי שניתן על ידי צוות הקורס בקובץ המצורף.
   יש למלא את הקוד המתאים במקומות הנדרשים (ולהוסיף שורות נדרשות, לפי הצורך)
   ii. Tree.h – תבנית מחלקה המייצגת כמעט עץ חיפוש בינארי. יש לממש כפי שמוגדר בטבלה מעלה, מותר להוסיף מתודות ו/או תכונות (במקרה הצורך.)
   iii. Node.h – תבנית מחלקה צומת בעץ. יש לממש כפי שמוגדר בטבלה מעלה, מותר להוסיף מתודות ו/או תכונות (במקרה הצורך).
   iv. קבצי קוד נוספים, במידה ותמצאו לנכון.

2. עליכם לשמור על כללי תכנות נאות כפי שפורסם במסמך במודל, ובפרט להקפיד על:

   i. אתחול משתנים ולתת להם שמות משמעותיים.
   ii. לא לכתוב ליטרלים בקוד (אלא להגדיר קבועים מתאימים).
   iii. פרט לקיצורים וראשי התיבות המופיעים במסמך, ראשי תיבות **לא** נחשב לשם בעל משמעות

3. עליכם להשתמש בדוגמאות מההרצאות והתרגולים. כמו כן אתם רשאים להשתמש בפונק' נוספות של טיפוסי ספריה הנ"ל שלא למדנו (לפי תקן c++98 בלבד, כמובן).

4. עליכם לתעד את הקוד שלכם באנגלית בלבד (אין צורך לדאוג לרמת האנגלית). יש לתעד לפי סגנון התיעוד שמופיע במודל.

5. אין בתרגיל זה דרישה של יעילות זמן.

6. הקדישו תשומת לב ומחשבה לאופן בו הפונקציות שלכם מקבלות פרמטרים .

7. הקדישו תשומת לב ומחשבה גם לאופן בו אתם מגדירים תבניות. **במיוחד** שימו לב שאתם משתמשים בצורה נכונה ברווחים בהגדרת התבניות.

8. יש להקפיד **שאין** דליפות זיכרון .

9. יש להקפיד כי פלט הקבצים זהה לחלוטין לפלט הניתן לכם (שימו לב שהפלט שניתן לכם הוא איחוד של שני ערוצי הפלט.)

10. יש להקפיד להשתמש ב streamהנכון לכל סוג של פלט(stderr stdout)

הוראות הגשה:

1. התרגיל להגשה בזוגות בלבד.
2. הקוד חייב להיכתב על פי מסכמות כתיבת הקוד (Coding Conventions) בקורס. קוד של עומד במסכמות לא יזכה במלוא הניקוד.
3. חובה לקמפל את התרגיל בעזרת סביבת Eclipse עם המהדר שהגדרנו לכם. תוכנית אשר תצליח להתקמפל בסביבה אחרת ולא בסביבה זו תחשב כקוד שלא עובר קומפילציה.
4. ההגשה חייבת להכיל קובץ ZIP יחיד בלבד (ולא קובץ RAR וכדומה) המכיל:
   - תיקייה בשם code ובה כל קבצי קוד המקור (h/cpp), ללא קבצי ההרצה (exe.)
5. שם הקובץ חייב להיות hw4_xxxxxxxxx_yyyyyyyyy.zip כאשר xxxxxxxxx ו-yyyyyyyyy הם מספרי תעודות הזהות של המגישים כולל ספרת ביקורת.
6. ההגשה היא אלקטרונית בלבד, דרך אתר ה-moodle של הקורס. תרגילים שיוגשו בכל דרך אחרת לא ייבדקו.
7. אין להגיש את אותו הקובץ פעמיים. התרגיל יוגש על ידי אחד מבני הזוג.
8. תרגיל בית של יוגש על פי הוראות ההגשה – לא ייבדק.
9. יש להקפיד על יושרת הכנת התרגיל וההגשה.
10. קוד שלא עובר קומפילציה יזכה לציון 0.

בהצלחה!