Data Structures for CS & ISE, Spring 2010

Assignment No. 3

Due date: 28.4.10

For questions, please use the "Help Forum" (http://myforum.bgu.ac.il/phpBB3/viewforum.php?f=133)

Assignment Topics

- AVL - Trees

תזמון פעולת מכונה

תיאור:
מפעל חדש נפתח בנגב, ובו מכונה ייחודית המבצעת פעולת עיצוב מתכות. כמובן שחברות רבות רוצות להשתמש במכונה, וקיים צורך לתזמן את השימוש בה. המפעל פנה למחלקה למדמ"ח ע"מ להקל על עצמו את המשימה. הדרישה היא לבנות מבנה נתונים שיתחזק את כל השעות שבהן מבקשים להשתמש במכונה . נדרש שמבנה הנתונים יתמוך בפעולות הכנסה , מחיקה וחיפוש של פעולה שחופפת לפעולה קיימת (ראה הגדרה של חפיפה, בסוף פיסקת התיאור).
לכל פעולה (TASK) יש שעת התחלה ושעת סיום, את שעת ההתחלה נסמן כ - beginning ושעת סיום כ – end.

הנחות מפשטות (מעט שונות מהמציאות):
אנו נניח בעבודה זו לשם פשטות, שהמכונה מתחילה לפעול בזמן 0 וזמן הפעולה שלה לא מוגבל (עד אינסוף) (למשל: יתכן שפעולה תתחיל בשעה 123)
ניתן להניח שכל שזמני ההתחלה והסיום הם מספרים טבעיים (שלמים ולא שליליים, אך כולל 0).
ניתן להניח שכל פעולה נמשכת לפחות שעה אחת.
שוב לשם פשטות: **לא** ייתכנו שתי פעולות שמתחילות באותו זמן.

חפיפה בין פעולות:
נגדיר ששתי פעולות חופפות אם אחת מתחילה תוך כדי משך השנייה (אחת מארבעת האפשריות הבאות):

מימוש:
מבנה הנתונים ימומש באמצעות **עץ חיפוש בינארי AVL**. כל קודקוד בעץ מתאים לפעולה אחת ומחזיק את נתוני הפעולה (שם, שעת התחלה, ושעת סיום). וכן שדה מיוחד נוסף MAX (הסבר בהמשך) ואת כל השדות שיש לקודקוד בעץ AVL רגיל.
המפתחות שעל פיהם ימוינו כל הפעולות הם שעת ההתחלה שלהם. זכרו, שעת ההתחלה של פעולה היא ייחודית, כלומר לא יתכנו שתי פעולות המתחילות באותה שעה.
ע"מ לממש את מבנה הנתונים יהיה עליכם לממש (בין היתר) את שתי המחלקות הבאות:

1. מחלקה Task:

String name;
int begin;
int end;

שדות הקשורים למידע על פעולת מכונה
השדה begin משמש גם כמפתח

2. מחלקה BinaryNode :
המחלקה תכלול את השדות:

Task task;
int max;
int height;
BinaryNode left;
BinaryNode right;
BinaryNodeparent;

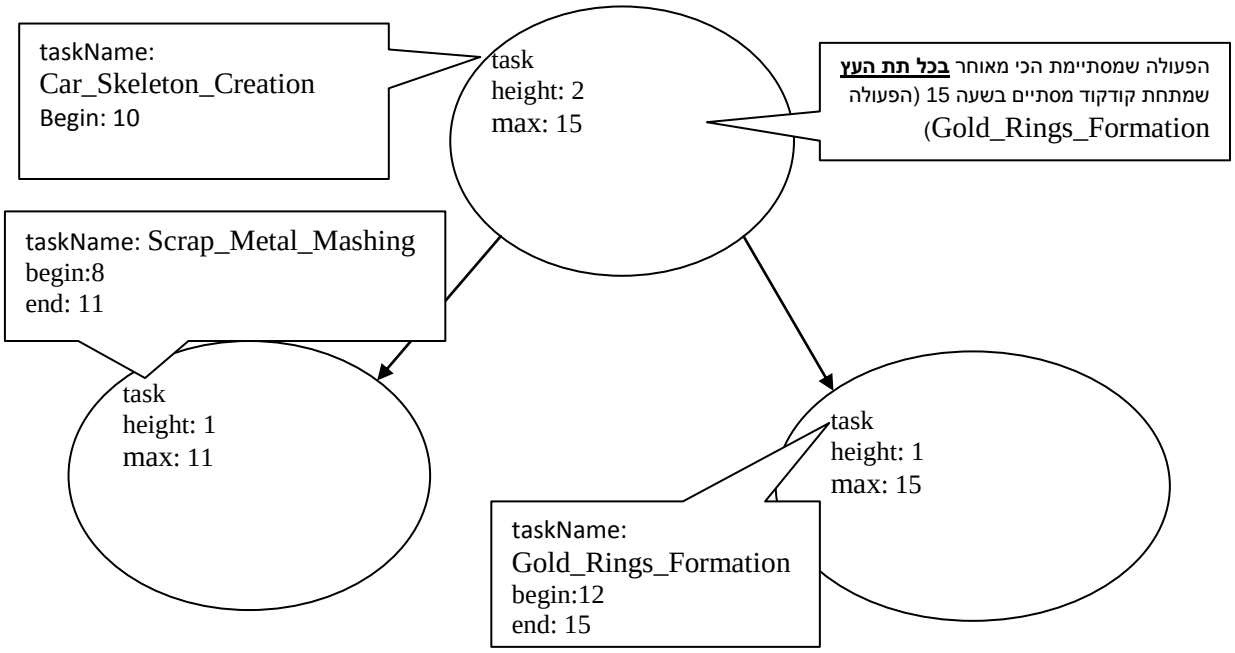
שדות שיש לכל עץ AVL

הסבר נוסף לגבי השדה max:
בשדה זה max שבקדקוד x נאחסן את שעת הסיום המאוחרת ביותר של **כל** הפעולות שמאוחסנות **בתת העץ** המושרש בקודקוד x.

דוגמאות:
דוגמא 1:
נניח כי המשימות שאנחנו רוצים לאחסן במבנה הנתונים הן:

NAME	BEGINNING	END
Gold_Rings_Formation	12	15
Car_Skeleton_Creation	10	13
Scrap_Metal_Mashing	8	11

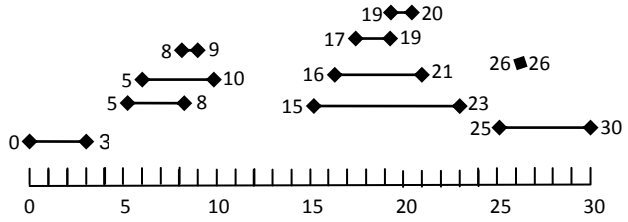
אזי מבנה הנתונים יכול להיראות כך:

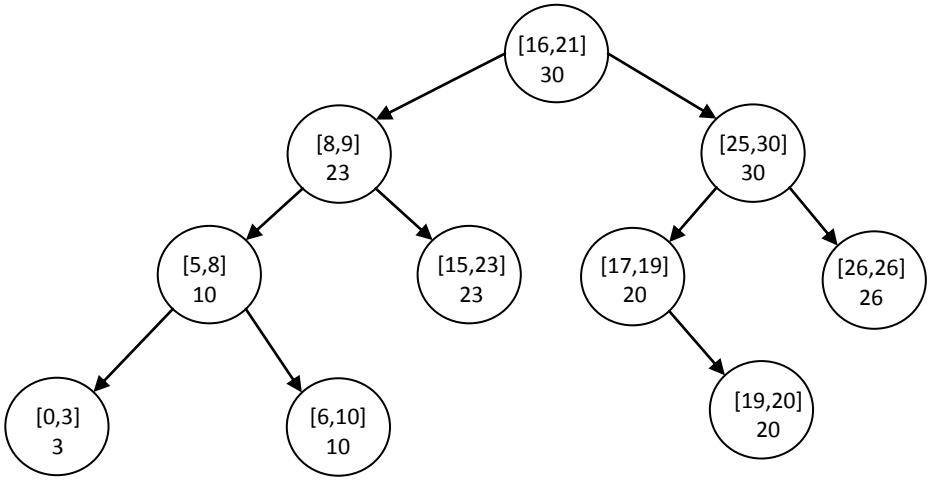


דוגמא 2:

הדוגמא השנייה הינה דוגמא לעץ שכולל יותר פעולות.
בציור של העץ בדוגמא הזו מצוינים עבור כל פעולה רק זמני ההתחלה והסוף שלה בסוגריים מרובעים וערך ה –max (כמובן שכל השדות האחרים בכל קודקוד מעודכנים אבל רק למען הפשטות ציינו רק את השדות לעיל).

הפעולות שקיימות בעץ (ללא שמות, לשם פשטות):





<u>הערות חשובות לפני שמתחילים</u>
הקוד בתרגיל מבוסס על הקוד אשר נלמד בקורס מבוא למדעי המחשב . לפני התחלת מימוש התרגיל **אנו ממליצים לכם בחום** לחזור על הרצאות 23-25 אשר נמצאות באתר מבוא למדעי המחשב 2010.
לעבודה זאת מצורף שלד קוד . יש להשלים את הקוד בקבצים המתאימים. **אין לשנות את מבנה השלד** – הבדיקה של העבודה מסתמכת על קבצים אלו וכל שינוי או מחיקה של קוד אשר סופק לכם יגרור הורדה בציון.
חשוב ביותר!!!! על מנת שהבדיקה האוטומטית תרוץ , אין לשנות את קובץ ה- MAIN של השלד. אם תרצו לשנות (בשביל להריץ בדיקות שלכם) את הקובץ, אתם רשאים לעשות זאת, כל עוד תקפידו <u>להגיש</u> את העבודה עם הקובץ MAIN המקורי ללא שינויים! כל שינוי בקובץ MAIN יגרור הורדת נק' אוטומטית.
בכל המשימות לא ניתן להניח דבר אשר לא מצוין במשימה.

וכעת לעבודה...

משימה 1: מימוש הממשק MyObject.
הממשק MyObject מיצג אובייקט שברצוננו לשמור בקודקודי העץ (שדה ה - Data) .
שימו לב כי ממשק זה מרחיב את הממשק Comparable , כלומר כל מחלקה אשר תממש אותו צריכה להיות ברת-השוואה.

public Comparable getKeyData() - מחזירה את המפתח שלפיו מתבצעת ההשוואה בעץ, כלומר את השדה אשר קובע את מיקום האיבר בעץ חיפוש. (במקרה שלנו, המפתח הוא זמן ההתחלה של הפעולה)

public Comparable getMaxData() - מחזירה את השדה שלפיו נקבע הערך המקסימלי בצומת. (במקרה שלנו, ערך זה הוא זמן הסיום של הפעולה)

public boolean overlap(Comparable start, Comparable end) - מחזירה true אם יש חפיפה בין this לבין start ו-end. חפיפה עבור MyObject מוגדרת בצורה דומה לחפיפה בין קטעים סגורים : הקטע הסגור של MyObject מוגדר ע"י [keyData, maxData] והקטע הסגור השני מוגדר ע"י [start, End] (אחד מהתנאים הבאים) :

- keyData <= start וגם maxData>=start
- keyData >= start וגם keyData <=end

ממשו את המחלקה Task אשר מממשת את הממשק MyObject.
שימו לב כי המחלקה Integer מממשת את Comparable.

משימה 2: מימוש המחלקות BinaryNode, BinaryTree.

המחלקה **BinaryNode** מייצגת קודקוד בעץ בינארי. למחלקה זאת השדות הבאים:

```
private BinaryNode left -  הבן השמאלי
private BinaryNode right - הבן הימני
private BinaryNode parent - האבא
private MyObject data - התוכן
private int height - גובה תת העץ ששורשו עצם המפתח
private Comparable max  - ערך maxData מקסימלי מכל תת העץ שמושרש בקודקוד
```

ממשו את השיטות והבנאים המסומנים ב //TODO .

המחלקה **BinaryTree** מייצגת עץ בינארי. למחלקה זאת השדה:

```
protected BinaryNode root - שורש העץ
```

ממשו את השיטות והבנאים המסומנים ב //TODO .

משימה 3: מימוש המחלקות BinarySearchNode, BinarySearchTree.

המחלקה **BinarySearchNode** מרחיבה את המחלקה BinaryNode ומייצגת קודקוד בעץ חיפוש בינארי . השלימו את השיטות הבאות :

.1

`protected BinaryNode insert(MyObject toAdd)` - הכנסת אובייקט חדש לתת העץ ששורשו עצם המפתח. על שיטה זו להחזיר את הצומת אליו נכנס האובייקט החדש.
אין להשתמש בבנאי המחלקה ליצירת קודקוד חדש אלא בשיטה:

```
protected BinarySearchNode createNode(MyObject data)
```

על שיטה זו לפעול ב (גובה העץ)O.

לא לשכוח לעדכן את השדה `height` ואת השדה `max` בקודקודים המתאימים.

.2

`protected BinaryNode remove(Comparable toRemove)` - הוצאת האיבר מתת העץ שמפתחו שווה ערך ל toRemove. על שיטה זו להחזיר את האבא של האיבר אותו אנו מסירים מהעץ . במידה ולא קיים איבר עם מפתח כנ"ל או האיבר שאותו אנו מסירים הינו השורש, על השיטה להחזיר null.
הערה: שימו לב כי אם ברצונכם למחוק קודקוד x שלו שני בנים, יש להחליף את x **בעוקב** שלו ולהמשיך במחיקה כפי שלמדתם.
לא לשכוח לעדכן את השדה `height` ואת השדה `max` בקודקודים המתאימים.
על שיטה זו לפעול ב (גובה העץ)O.

.3

`protected MyObject overlapSearch(Comparable start,Comparable end)` - על השיטה למצוא איבר בתת העץ אשר חופף ל start , end כפי שמוגדר במשימה 1. במידה ויש יותר מ פעולה אחת כזו, תוחזר הראשונה בה פוגשים אם פועלים ע"פ סריקה של **עץ חיפוש בינארי.**
במידה ולא קיים איבר כנ"ל, על השיטה להחזיר null .
על שיטה זו לפעול ב (גובה העץ)O.

רמז: יש להיעזר בשדה max.

המחלקה BinarySearchTree מרחיבה את המחלקה BinaryTree ומייצגת עץ חיפוש בינארי.
ממשו את השיטות והבנאים המסומנים ב //TODO .

משימה 4: מימוש המחלקות AVLSearchNode, AVLSearchTree .

המחלקה **AVLSearchNode** מרחיבה את המחלקה BinarySearchNode ומייצגת קודקוד בעץ חיפוש AVL.

ממשו את השיטה: `protected BinaryNode insert(MyObject toAdd)`
אשר מכניסה אובייקט לתת עץ ה AVL. שימו לב כי שיטה זו משתמשת בהכנסה של עץ חיפוש בינארי , שלאחר ההכנסה צריך לבצע רוטציות. על השיטה להחזיר את הקודקוד אליו נכנס האיבר.
חובה לממש את שיטה זו ב O(log(n))!

המחלקה **AVLSearchTree** מרחיבה את המחלקה BinarySearchTree ומייצגת עץ חיפוש AVL.
השלימו את השיטות הבאות :

```
public void insert(MyObject toAdd)
```

הכנסת איבר לעץ AVL . חובה לממש את שיטה זו ב O(log(n)) !!

```
public void remove(Comparable toRemove)
```

הוצאת איבר אשר מפתחו שווה ערך ל toRemove . חובה לממש את שיטה זו ב O(log(n)) !!
לא לשכוח לעדכן את השדה `height` ואת השדה `max` קודקודים המתאימים.

<u>משימה 5</u>: בדיקות

המחלקה **Main** מכילה פונקציית main . הפונקציה קוראת ערכים מקובץ משתמשת במחלקות אשר בניתם, יוצרת עץ AVL מבצעת עליו פעולות ומחזירה את העץ. בדקו את התוצאות שלכם בהתאם לפלט הרצוי.

הפעולות הן : insert, delete, overlap, inorder and byLevels

שתי הפעולות האחרונות מדפיסות את העץ (האחת ב inOrder והשנייה לפי שכבות העץ) ע"י שימוש בשיטות שסופקו לכם הנמצאות במחלקה BinaryTree

---

את הקבצים שאתם מגישים יש לכווץ לקובץ zip (ללא ספריות) ששמו הוא ת .ז המגישים, ואותו לעלות למערכת ההגשה.

<u>מומלץ לעבוד לפי סדר המשימות כפי שהוגדרו , כך עבודתכם תהיה יעילה ומהירה יותר . לפני שעוברים למשימה הבאה, בדקו בצורה טובה את מימוש המשימה הקודמת!</u>

<u>בהצלחה!</u>