# בס"ד

# תרגילי ריצה -תשפ"ה

**הערות כלליות לגבי כל תרגילי הריצה:**

* **יהיו 5 תרגילי ריצה במהלך הסמסטר, ויש להגיש 4 מתוך 5. תרגיל שלא יוגש בזמן, ציונו יהיה אפס.**
* **בכל אחד מהתרגילים יש להגיש תרגיל מקורי שפותר את הדרישות כפי שהוגדרו בדף התרגיל של תשפ"ה.**
* **תרגיל שלא יעבור קומפילציה או תרגיל ש"עף" בזמן ריצה או תרגיל שאינו בנוי במדויק על פי דרישות התרגיל תשפ"ה יקבל ציון 0 בלי אפשרות להגשה מחדש לאחר תאריך ההגשה.**
* **שמות ות.ז. המגישים חייבים להופיע בראש הקובץ שבו נמצא ה-main.   
  מגיש ששמו לא יופיע בתוך ה-main של התרגיל, ציונו בתרגיל יהיה 0 בלי אפשרות לתיקון.**
* **בראש קובץ ה-main יש לציין את הבונוסים שנעשו בתרגיל, אם נעשו.**
* **תרגיל אמור להיות כתוב בצורה מובנית, עם הקפדה על תיעוד (הערות), עימוד מתאים, חלוקה מתאימה לקבצים, ובאופן כללי כתיבה ברורה, מודולרית ומסודרת. כל הנקודות האלו צפויות להיבדק במהלך הבדיקה הידנית.**
* **חלק מהמבחן הסופי מבוסס על ידע שנרכש תוך כדי ביצוע תרגילי הריצה.**
* **פרט למקרים של חופשת לידה ומילואים, לא תינתן אפשרות להארכת תקופת ההגשה המופיעה במודל. על מנת לאפשר לסטודנטים שמסיבה כלשהי (שאינה לידה ומילואים) לא הספיקו להגיש במועד ההגשה הרגיל, תהיה תיבת הגשה באיחור בניכוי 20 נקודות מהציון.**
* **בגמר ההגשה של כלל התרגילים, הבודק האוטומטי של מודל מזהה העתקות, וציוני התרגילים המועתקים, כולל המקור וכולל כל ההעתקים, יהיה אפס, ללא יכולת להגיש על כך ערעור וללא יכולת להגיש מחדש.**
* **אין לשתף/להציג פתרון או חלק מפתרון לתרגיל בשום מדיה שהיא (כולל בעל פה). תלמיד שיתברר ששיתף פתרון או שהשתמש בפתרון ששותף איתו, שמו יועבר לוועדת משמעת, וציונו בתרגיל יהיה אפס. למען הסר ספק, פתרון תרגיל עם סטודנט שאינו בן הזוג, דינו כשיתוף פתרון. למען הסר ספק, שאלה "איך פתרת את סעיף..." או מענה לשאלה כזו, דינם כשיתוף פתרון.**

# תרגיל 1: עץ רב בנים - מערכת קבצים

## רקע לתרגיל

מטרות תרגילי התכנות בקורס 'מבני נתונים' הן

* הטמעת החומר הנלמד בקורס, תוך מימוש מבני נתונים חדשים
* שילוב מבני נתונים שונים שנלמדו בקורס 'מבני נתונים א' למבני נתונים מורכבים היכולים לתת מענה לבעיות מתקדמות
* פיתוח של מבני נתונים גנריים שתומכים במגוון רחב של תרחישים
* התמודדות עם פיתוח מערכות בקנה מידה בינוני ועם האתגרים הקשורים אליהן

בתרגיל 1 נשלב שני מבני נתונים שנלמדו בקורס 'מבני נתונים א': עץ ורשימה לקבלת מבנה נתונים מתקדם: רשימה של עצים רבי בנים. השילוב בא לידי ביטוי בשני אופנים שונים: תוכן האיברים ברשימה הראשית יהיו עצים והבנים של קודקודי העץ ינוהלו ברשימה. בנוסף, על מנת שמבני הנתונים שתפתחו יהיו רב-שימושיים, נגדיר את מבנה הנתונים בצורה גנרית, כך שבעתיד תוכלו להשתמש בהם גם כדי לתמוך בנתונים מסוגים שונים.

## תאור כללי

בתרגיל זה נבנה מבנה של מערכת קבצים המכילה קבצים ותיקיות, כאשר תיקיות יכולות להכיל קבצים ו/או תת-תיקיות.

נרצה לאפשר את הפעולות הבאות:

* הוספת קבצים/תיקיות חדשים למערכת
* הוספת קבצים/תת-תיקיות לתיקייה קיימת
* מחיקת קבצים/תיקיות (כולל מחיקת כל התוכן שלהם)
* הדפסה של מבנה התיקיות והקבצים בצורה מעומדת (עם הזחות מתאימות בין תת-תיקיה/קובץ לבין התיקיה המכילה אותם)
* עדכון תוכן של קובץ (הוספת/מחיקת טקסט)
* הדפסה של תוכן קובץ

## ממשק המערכת

על מנת לתמוך בפעולות, עליכם לממש מחלקה בשם FileSystem המכילה את הפונקציות הבאות:

void addTree(NodeRecord<string>\* rootRecord);

void addRecord(const string& parentKey, NodeRecord<string>\* record);

void deleteRecord(const string& key);

void deleteTree(const string& rootKey);

void printFileSystem() const;

void printFileContent(const string& fileName) const;

void appendToFile(const string& fileName, const string& content);

void deleteSubstringFromFile(const string& fileName, const string& substring);

## הנחיה למימוש המערכת

הערה מקדימה: ההנחיה מהווה הצעה לפתרון התרגיל באופן כללי העומד בעקרונות עיצוב תכנה. מטרת ההנחיה היא לספק לכם שלד לתוכנה שמממשת את מבני הנתונים בצורה כללית, כך שהיא תומכת במגוון רחב של יישומים ולא רק בדרישות התרגיל המסוים. שימו לב כי השלד אינו מכיל את כל הפונקציות הנדרשות ושתצטרכו להוסיף פונקציות עזר משלכם. אנו מקווים כי תלמדו מההנחיה כיצד כדאי לגשת לפתרון בעיות פיתוח ושתחשפו לחשיבה גנרית. עם זאת **אין חובה להשתמש בהנחיה** ותוכלו לבחור לפתור את התרגיל בדרך אחרת מזו המוצעת כאן. הדרישות המינימליות הן לממש את ממשק המערכת המתואר לעיל באופן שיעמוד בדרישות של הבודק האוטומטי. קחו בחשבון כי התוכנה שלכם תיבדק גם על ידי בודקים ידניים שיבדקו גם סגנון קוד, הערות ומבנה.

נייצג את מבנה הנתונים התומך במערכת הקבצים באמצעות רשימה של עצים רבי בנים גנריים, כאשר בתרגיל שלנו בכל קודקוד של עץ יש תיקיה או קובץ. קודקודים שמכילים תיקיות מצביעים לרשימת בנים והבנים של כל קודקוד הם תתי התיקיות והקבצים שמכילה תיקייה זו. הקודקודים שמכילים קובץ או תיקייה שאין בה קבצים או תיקיות יהיו עלי העץ. תוכן הקודקודים יהיה מזוהה באמצעות מפתח (במקרה שלנו: שם התיקיה או הקובץ)

נתאר את המחלקות המרכיבות את המערכת באופן bottom-up (כלומר מהפרטים הפשוטים אל המערכות המורכבות)

המערכת תכלול את המחלקות הבאות המייצגות את הנתונים שיאוחסנו בקודקודי העץ:

1. קובץ (File) המייצג קובץ טקסט  
   **כל קובץ יכיל את השדות הבאים:**
2. name: שם הקובץ (מחרוזת)
3. content: תוכן הקובץ (מחרוזת)

**כל קובץ יכיל את הפעולות הבאות:**

1. בנאי: יוצר קובץ ריק עם שם נתון
2. הורס: מחיקת הקובץ.
3. appendContent: הוספת טקסט לסוף תוכן הקובץ
4. deleteSubstring: מחיקת המופע הראשון של המחרוזת הנתונה מתוכן קובץ
5. תיקיה (Directory) המייצגת תיקיה במערכת  
   **כל תיקיה תכיל את השדות הבאים:**
6. :name שם התיקיה הנוכחית. (מחרוזת)

**כל תיקיה תכיל את הפעולות הבאות:**

1. בנאי: יוצר תיקיה ריקה עם שם נתון
2. הורס: מחיקת התיקיה
3. נרצה לאפשר לקבצים ולתיקיות להיות מאוחסנים בעץ רב בנים. לצורך כך נגדיר מחלקה מופשטת וגנרית NodeRecord שתייצג תוכן של קודקוד בעץ. **משתנה הטיפוס T במחלקה הגנרית ייצג את הטיפוס של המפתח של הרשומה**. המחלקות קובץ ותיקיה יירשו מהמחלקה NodeRecord ויממשו את הפעולות שלה, כאשר המפתח שלהם הוא מטיפוס מחרוזת.  
   **כל רשומה תכיל את הפעולות הבאות:**
4. הורס: מחיקת הרשומה
5. getKey: מחזירה את המפתח של הרשומה מסוג T. קובץ ותיקיה יממשו את הפונקציה על ידי החזרת השם שלהן.
6. getType: מחזירה מחרוזת המייצגת את הטיפוס של הרשומה. מימוש הפונקציה בקובץ יחזיר את המחרוזת "File". מימוש הפונקציה בתיקיה יחזיר את המחרוזת "Directory". מטרת הפונקציה היא לאפשר להבדיל בין הטיפוסים השונים שמוכנסים לעץ בזמן ריצה.
7. canHaveChildren: פונקציה בולאנית עם מימוש ברירת מחדל שמחזיר true. המחלקה קובץ תדרוס את הפונקציה במימוש שמחזיר false כדי למנוע הוספת קודקודים מתחת לקבצים.

לאחר שהגדרנו את הרשומות, נעבור למימוש מבנה הנתונים רשימה של עצים רבי בנים. גם כאן נתאר את המבנה באופן bottom-up.

1. קודקוד (MultiTreeNode) גנרי עם משתנה טיפוס T

**כל קודקוד יכיל את השדות הבאים:**

1. רשומה עם משתנה טיפוס T
2. רשימת (list) בנים מסוג קודקוד עם משתנה טיפוס T

**כל קודקוד יכיל את הפעולות הבאות:**

1. בנאי: יוצר קודקוד עם רשומה נתונה וללא בנים
2. הורס: מחיקת הקודקוד (כולל שחרור זכרון הרשומה)
3. הוספת בן: במידה והרשומה בקודקוד הנוכחי יכולה להכיל בנים, הפונקציה תוסיף את הקודקוד הנתון לרשימת הבנים
4. הסרת בן: בהנתן מפתח, אם יש בן עם מפתח זה, יש להסיר אותו מרשימת הבנים ולדאוג לשחרור הזיכרון שקשור אליו.
5. הדפסה: הדפסת מפתח הרשומה ששמורה בקודקוד
6. עץ (MultiTree) גנרי עם משתנה טיפוס T

**כל עץ יכיל את השדה הבא:**

1. שורש: קודקוד עם משתנה טיפוס T

**כל עץ יכיל את הפעולות הבאות:**

1. בנאי: יוצר עץ עם שורש שמכיל את הרשומה הנתונה
2. הורס: מחיקת העץ (כולל שחרור זכרון של כל הקודקודים).
3. חיפוש רשומה: בהנתן מפתח מטיפוס T הפונקציה תחזיר מצביע ל-Record שהמפתח שלו שווה למפתח המבוקש (או nullptr אם אין רשומה עם מפתח מתאים בעץ). במערכת שלנו נקרא לפונקציה עם מחרוזת שתייצג את שם הקובץ/תיקיה.
4. הוספת רשומה מתחת לקודקוד קיים: בהינתן מפתח של רשומה שמאוחסנת בעץ ורשומה חדשה, הפונקציה תחפש את הקודקוד המתאים בעץ ותוסיף לו בן עם הרשומה החדשה. אם הקודקוד לא נמצא בעץ, לא יתבצע דבר.
5. מחיקת רשומה: בהינתן מפתח של רשומה שמאוחסנת בעץ, הפונקציה תחפש את הקודקוד המתאים בעץ ותמחק אותו (כולל שחרור זכרון של תת העץ המושרש בו). הערה: לא ניתן למחוק את שורש העץ באמצעות פונקציה זו.
6. הדפסה: הדפסת המפתחות של הרשומות שמאוחסנות בתת העץ בצורה היררכית (עם הזחה של 3 רווחים לכל רמה).
7. רשימת עצים (TreeList) גנרי עם משתנה טיפוס T

**כל רשימת עצים תכיל את השדה הבא:**

1. רשימה (list) של עצים

**כל רשימת עצים תכיל את הפעולות הבאות:**

1. בנאי: יוצר רשימה ריקה
2. הורס: מחיקת הרשימה (כולל שחרור זיכרון).
3. חיפוש רשומה: בהנתן מפתח מטיפוס T הפונקציה תחזיר מצביע ל-Record שהמפתח שלו שווה למפתח המבוקש (או nullptr אם אין רשומה עם מפתח מתאים באף אחד מהעצים ברשימה).
4. הוספת עץ חדש: בהינתן רשומה, הפונקציה תוסיף לסוף הרשימה עץ חדש ששורשו יכיל את הרשומה.
5. הוספת רשומה לעץ קיים: בהינתן מפתח של רשומה שמאוחסנת ברשימת העצים ורשומה חדשה, הפונקציה תחפש את הקודקוד המתאים בעצים ותוסיף לו בן עם הרשומה החדשה. אם הקודקוד לא נמצא באף עץ, לא יתבצע דבר.
6. מחיקת רשומה: בהינתן מפתח של רשומה שמאוחסנת ברשימת העצים, הפונקציה תחפש אותה ותמחק אותה מהעץ המתאים.
7. הדפסה: הדפסה של כל העצים ברשימת העצים.

**לאחר מימוש מבני הנתונים תוכלו להשתמש במחלקה רשימת עצים למימוש המחלקה FileSystem.**

**מצורף במודל קובץ main וכן קבצי ה header של מבני הנתונים. קובץ ה main ומאפשר לבצע שוב ושוב (באמצעות קלט שנמצא בקובץ קלט) את כל אחת מהפעולות שהוגדרו בתרגיל. קובץ ה main אינו תומך בקלט הכולל רווחים. אין לשנות את קובץ ה main פרט להוספת פרטי המגישים.**

## הערות כלליות לגבי בדיקת התרגיל

* תקינות קלט: ניתן להניח שהקלט לפונקציות תקין. יחד עם זאת ייתכן כי בחלק מהמקרים תהיה התייחסות למפתחות שאינם נמצאים בעץ או נסיון להוסיף בנים מתחת לקודקוד שמכיל קובץ. במקרים כאלה הפונקציה צריכה להסתיים באופן תקין ללא שהשתנה דבר במבנה העץ.
* סדר ההדפסות והחיפוש: ההכנסה לרשימה יהיה תמיד לסוף הרשימה. באותו אופן, הוספת בן לקודקוד תוסיף אותו לסוף רשימת הבנים. סריקת העץ (בעת הדפסה או חיפוש קודקוד) תתבצע בסדר pre-order, כלומר שקודם נסרוק את שורש תת העץ ואחר כך נסרוק את תתי העצים שלו לפי סדר ההופעה של הבנים ברשימת הבנים.
* הופעה של אותו מפתח מספר פעמים בעץ: במקרה של פעולה המתייחסת למפתח שקיים בעץ (לדוגמא כאשר רוצים להוסיף קובץ חדש מתחת לתיקיה בשם מסויים או כאשר רוצים למחוק תיקיה בשם מסויים), אם המפתח מופיע יותר מפעם אחת בעץ, נבצע את הפעולה על הקודקוד הראשון שמצאנו עם המפתח.

**בהצלחה רבה!**

## נספח: תרשים UML

על מנת לסייע לכם להבין את המבנה של הצעת הפיתרון, אנו מצרפים תרשים שמתאר אותו. התרשים כולל את המחלקות שבמערכת ואת הקשרים ביניהם. בהמשך התואר תלמדו באופן מפורט את המשמעות של כל אחד מחלקי התרשים, וכרגע תשתמשו בו ככלי רעיוני.

