**

**

*נקודות חשובות*

* *חלק יבש*
  + *כשמוחקים אמן – אכן mlogn*
    - *למה – כי אנחנו רצים על המערך של השירים של אותו אמן, מוצאים לפי הסדר בעץ ה-AVL של האמנים ולמחוק את כל השירים שם – זה מסתכם ל-mlogn+m (צריך להסביר למה)*
  + *לוודא ולהסביר שאר ענייני סיבוכיות*
  + *לכתוב על טיפול בשגיאות ביבש*
  + *לציין את הפעולות בפירוט*
* *ווידוא ענייני C++*
  + *המרות*
    - *לוודא אם מותר מפורשות או שצריך STATIC*
  + *חריגות?*
    - *BAD\_ALLOC?*
    - *האם פספסנו משהו במימוש הנוכחי? (שלא מימשנו כ-exceptions)*
* *לטפל בקוד-*
  + *מימוש הרשימה*
  + *למחוק StreamList*
    - *חשבתי לממש באופן שמוחקים NODE-NODE – האם עומד בסיבוכיות?*
    - *צריך לעשות מתודה נוספת שמוצאת האם ה-key קיים?*
  + *יצירת StreamListNode*
    - *כרגע יוצר NODE רק אם יש לי עץ – לשנות?*
  + *Const ב-NODE – זה אומר שאי אפשר לשנות NODE, וצריך לאפשר את זה*
  + *להפוך פונקציות VOID לפונקציות ENUM?*
  + *לקבל דברים by value או by reference?*
  + *לשים לב לשימוש בבנאים והורסים – האם משתנה שהגדרתי יימחק אוט' בסיום הבלוק?*
    - *לראות שהשתמשנו נכון ב-new וב-delete*
  + *Include אחד לשני – artist-list*
    - *נכון להשתמש בהצהרות לפני שבכלל עשיתי include?*
    - *ענייני מערך של מצביעים בתוך מחלקה – עבור מערך השירים*
  + *לעבור ולסדר את ההערות והפונקציות – לוודא שהכל כמו שצריך*
    - *לראות אם יש פונקציות שאנחנו בכלל לא צריכים ופוגעות בנו*
  + *מה משמעות קובץ ה-main?*
  + *דברים שלא עובדים*
  + *טסטים*
    - *רשמיים*
    - *של אנשים*
    - *שרת*
  + *בדיקות אחרות*
    - *קונבנציות?*
    - *הערות?*

*פונקציות נדרשות-*

*Init*

*AddArtist*

*RemoveArtist*

*AddSongToCount*

*NumberOfStreams*

*GetRecommendedSongs*

*Quit*

*צריך לבדוק-*

* *אלגוריתמי חיפוש בעצי ה-AVL ועמידה בסיבוכיות*
* *מערך שירים – מכיל פוינטרים או ~~מחלקה מסוג SONG~~?*
* *עץ AVL של מספר השמעות – מכיל פוינטר*

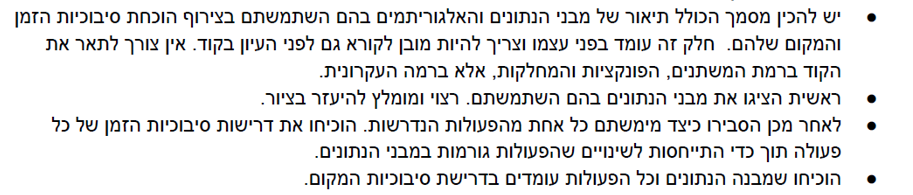
*מבנה הנתונים יורכב מהחלקים הבאים:*

*מחלקות כלליות*

1. *מחלקה גנרית של עץ AVL*
   1. *תומכת בכל הפעולות שצריך לעשות בעץ-*
      1. *הוספת צומת*
      2. *הוצאת צומת*
      3. *חיפוש צומת*
      4. *צריך לבנות נכון את האלגוריתמים?*
         1. *צריך לחשוב איך עושים את זה?*
   2. *כוללת גם NODE כללי גנרי*
      1. *שדות*
         1. *מצביע תבניתי ל-info (artist, song…)*
         2. *מצביע לבן ימני*
         3. *מצביע לבן שמאלי*
2. *מחלקה של אמן*
   1. *שדות:*
      1. *Int – Artist\_id? או שזה יהיה שם האמן?*
      2. *מערך – כלל השירים (מערך של מצביעים?)*
      3. *עץ AVL – מספר ההשמעות שיש לכלל השירים*
   2. *מתודות:*
      1. *Ctor*
         1. *בונה את השדה artist\_id דיפולטיבית*
         2. *בונה מערך בגודל N (מספר השירים בעת יצירת ה-Artist)*
            1. *הקצאה דינמית*
            2. *מכניס כפוינטר לרשימת ההשמעה 0*
         3. *בונה אובייקט מסוג AVL מסוג מספר השמעות?*
            1. *בהתחלה עם 0 צמתים (NULL)*
      2. *Dtor*
         1. *מוחק את השדה artist\_id דיפולטיבית*
         2. *מוחק את המערך*
            1. *מחיקה דינמית – delete*
            2. *צריך לדאוג למחיקה של הפוינטר שברשימה המקושרת המתאימה*
         3. *מוחק את האובייקט מסוג AVL*
            1. *מחיקה ע"י המתודה של AVL*
      3. *Cctor – למחוק (שלא תהיה אפשרות)*
      4. *Assign op – למחוק (שלא תהיה אפשרות)*
      5. *Getters*
         1. *Get\_song*
            1. *מחזיר SONG לפי אינדקס*
         2. *Get\_id*
            1. *מחזיר את ה-artistid*
      6. *Editors*
         1. *Add\_play\_to\_song*
            1. *מעלה ב-1 את מספר ההשמעות בשיר לפי אינדקס*
            2. *או*
            3. *מעביר את המצביע*
3. *מחלקה של שיר*
   1. *כנראה שמספיק להחזיק פוינטר – או שנעשה מחלקה (צריך לציין שדות ומתודות)*
4. *מחלקה של מספר השמעות?*
   1. *מספיר שה-key יהיה המספר, והאיבר עצמו יהיה עץ AVL של שירים (של INT המייצג SONGID)*

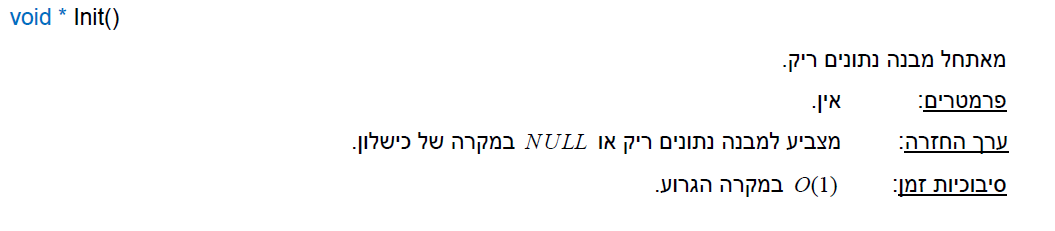
*מבנה הנתונים עצמו*

1. *X משתנים כלליים עבור מבנה הנתונים:*
   1. *שדכ*
2. *X – עץ AVL כללי של אמנים*
   1. *קיים אחד כזה באופן כללי*
   2. *עץ הבנוי מ-NODES מסוג artist*
3. *X – עצי AVL של מספר ההשמעות*
   1. *קיים אחד כזה עבור כל artist*
   2. *עץ הבנוי מ-NODES מסוג song*
4. *X – עצי AVL של שירים*
   1. *קיים אחד כזה עבור כל מספר השמעה*
   2. *כל צומת – SONGID*
5. *X – רשימה מקושרת של "מספר השמעות"*
   1. *נוצרת בעת האתחול הראשוני init*
   2. *קיימת אחת כזו*
   3. *מכילה איברים מסוג עץ AVL מסוג Artist*
      1. *(שיש להם את אותו מספר השמעות לפי האיבר)*
6. *דגע*

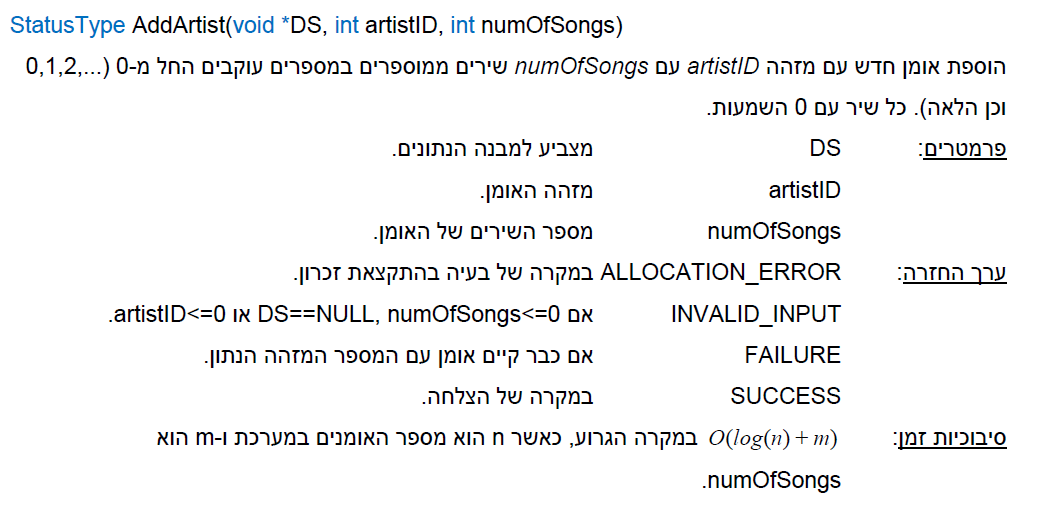
**

***מימוש כל אחת מהפעולות הנדרשות***

*Init*

**

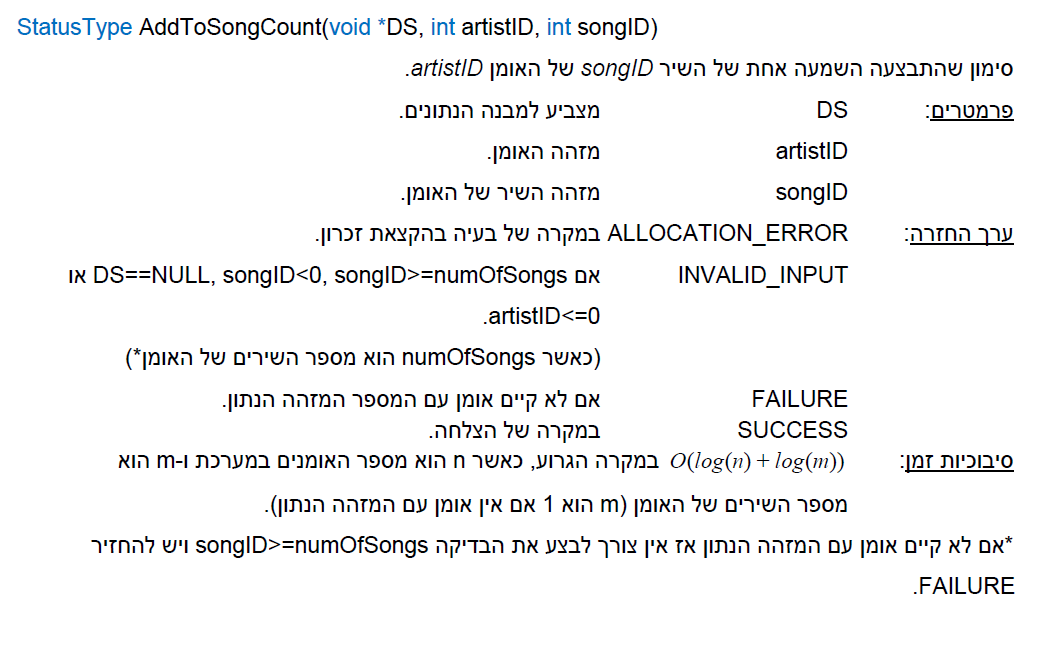
*AddArtist*

**

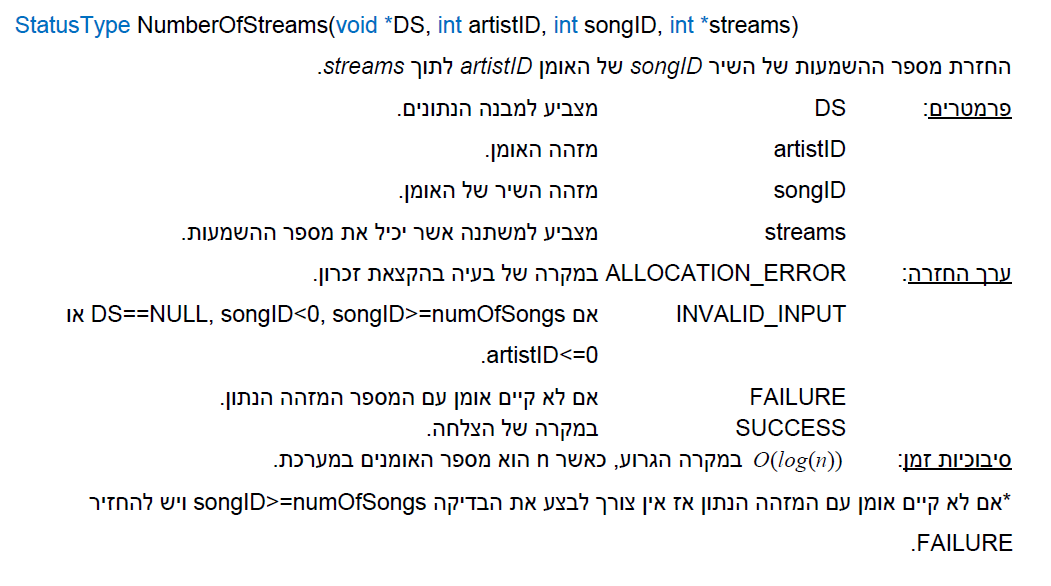
*RemoveArtist*

**

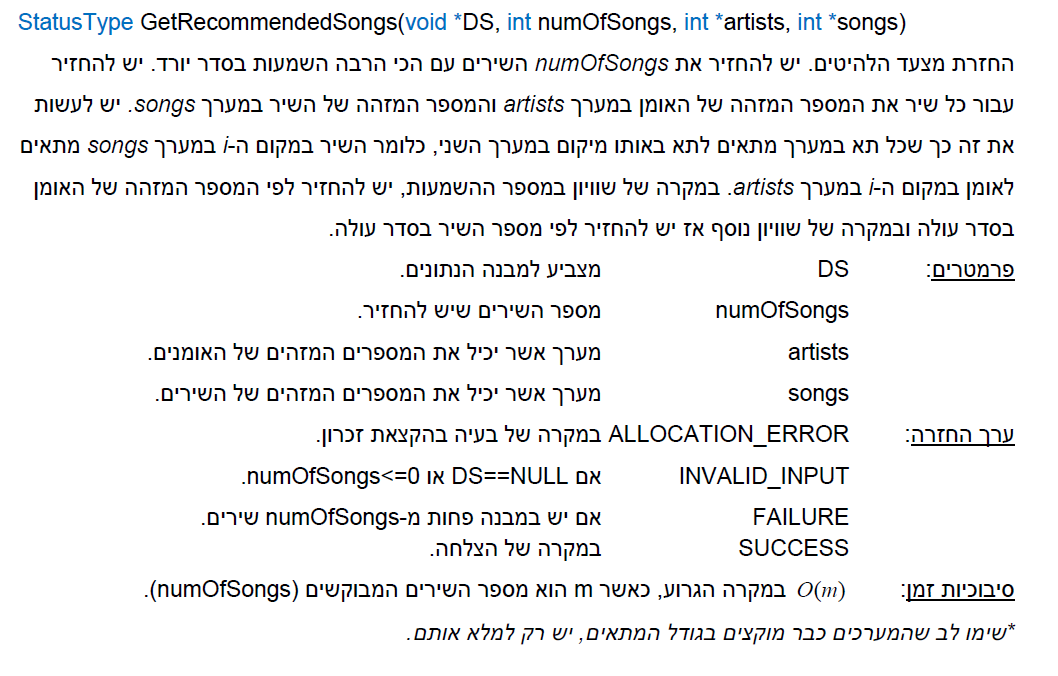
*AddSongToCount*

**

*NumberOfStreams*

**

*GetRecommendedSongs*

**

*Quit*

*ציור המבנה:*

עץ אמנים:

**אמן 1:**

artist\_Id: 1

Total\_num\_of\_songs: 2

full\_songs\_list:

num\_streams\_tree

*טענת עזר שנתשמש בעת הוכחה על עמידה בסיבוכיות הנדרשת: נראה שבמידה ויש לך עץ ואתה מתחיל בצומת בעל האינדקס הכי קטן בעץ, ואתה עובר(מדפיס) לפי סדר r צמתים, אז תעשה סך הכל O(r) פעולות.*

*הוכחה: מכיוון שאנחנו מתחילים בצומת המינימאלי, הצומת הזו נמצאת בתחתית העץ (העלה הכי שמאלי) וכדי להגיע לצומת ה-r נעשה (מספר העליות שביצענו בעת המעברים)O, עתה מכיוון שאנחנו עוברים לפי סדר בכל עלייה כזו נצטרך או שלהדפיס את הצומת (במידה ועלית מבן שמאלי) או שלא (עלייה מבן ימני) ואז כבר הדפסת את הצומת הזו (כי הבן הימני יותר גדול מהצומת הנוכחי ואנחנו עוברים לפי סדר). לכן מספר העליות שסך הכל ביצענו הוא O(r), ולכן סך הפעולות שנעשה הוא O(r). כפי שרצינו להראות.*

*Init:*

*באתחול של מבנה הניתונים שלנו, אנחנו יוצרים עץ של אמנים ריק וזה קורא ב-O(1), בנוסף יוצרים רשימה מקושרת, עם צומת אחת המתאימה לצומת של 0 השמעות. בצומת זו יש לנו עץ אמנים ריק, ולכן יצריתה היא O(1). עתה יצירת רשימה מקושרת והכנסת איבר אחד בלבד אליה היא O(1), ולכן אנחנו עונים על תנאי הסיבוכיות בשאלה.*

*GetRecommendedSongs:*

*נראה שהפונקציה אכן עונה על התנאי של הסיבוכיות שהוא O(m). אנחנו מתחילים בצומת האחרון ברשימת ההשמעות שלנו (מכיוון שיש לנו פוינטר אליו ההגעה לצומת היא O(1))שהיא מייצגת את מספר ההשמעות המקסימלי שיש לנו במערכת (נסמנו M) בצומת זו יש לנו עץ של פוינטרים לעצים המתאימים לM שירים לאומנים אשר יש להם לפחות שיר אחד עם M השמעות. אנחנו ניגשים לאיבר הראשון בעץ הפוינטרים (מכיוון שיש לנו פוינטר first תחת ה-AvlTree זה נעשה ב-O(1))- זה בעצם פויטר לעץ המתאים של האומן עם האינדקס הכי קטן שיש לו M השמעות (נסמנו ) . בעץ של השירים עם M השמעות תחת נעבור את השירים לפי הסדר (שוב מתחילים מהשיר עם האינדקס הכי קטן,הגעה אליו זה O(1) בגלל first ), כאשר על כל שיר שנעבור נשים את הערכים המתאימים למערכים, מכייון שיש לנו count שמתחיל מ-0 ושאותו אנחנו מגדילים ב-1 לאחר כל מעבר על שיר נדע תמיד את האינדקס המדוייק שבו אנחנו צריכים לשים את הערכים במערך. אנחנו נעבור על כל העץ (במקרה וnumSongs גדול ממספר השירים שיש ל- עם M השמעות ) או שנעבור רק על חלק מהעץ. נסמן את מספר הצמתים בעץ שנעבור עליהם ב-**. מטענת העזר סך הפעולות שנעשה הוא O(). וזה נכון לכל העצים שלעיהם נעבור. מכיוון שאנחנו רצים בלולאות while עם count שמגדילים אותו ב-1 לאחר כל מעבר ומכיוון שמעבר מצומת לצומת הקודמת ברשימת השמעות היא O(1) על שיר נעשה סך הכל מספר פעולות של:*

*.*

*כנדרש, כאשר השוויון הראשון נובע מטענה שעשינו בתרגול.*

*Quit:*

*תחילה נשים לב שתמיד- ולכן .*

*ב-quit מה שאנחנו עושים בעצם זה לקרוא לדסטרטור של ה-MusicManager. שהוא קורא לדיסטרקטור של של העץ אמנים שלנו ולדיסטרקטור של הרשימת השמעות. נראה עבור כל אחד שהוא עומד בתנאי הסיבוכיות.*

*עץ אמנים:*

*בעץ אמנים יש לנו עץ ובו n צמתים (כמספר האמנים), נעשה זום אין לצומת כזו, הצומת הזו מייצגת אומן מסויים (), באומן הזו יש עץ השמעות(num\_of\_streams\_tree) כאשר ה-key של כל צומת זה מספר ההשמעות וה-element של כל צומת זה עץ עם השירים שיש לאומן המסויים עם מספר השמעות של key.*

*נצטרך למחוק את העץ שירים הזה, המחיקה של עץ כזה היא O( כאשר זה מספר השירים שיש לאמן עם השמעות, בנוסף בסופו של דבר נצטרך למחוק את המערך full\_songs\_list ומכייון שזה בסופו של דבר פוינטר מחיקתו זה ב- (גודל המערך)O (מחיקת האיברים שהוא מצביע אליהם נעשית דרך המחיקה של הרשימה המקושרת שלנו). לכן סך הכל עבור מחיקה של אמן מעץ האמנים שלנו נעשה מס' פעולות שהוא*

*כאשר זה מספר השירים שיש לאמן a. זה עבור אמן ספציפי, עבור מחיקה של כל עץ האמנים נעשה באותה דרך:*

*.*

*רשימת השמעות:*

*נצטרך לעבור צומת צומת ובכל צומת למחוק את הצמתים בעץ הפוינטרים המתאים למספר ההשמעות של הצומת בו אנחנו נמצאים (את הדברים אליהם אנחנו מצביעים אנחנו מוחקים בעת מחיקת עץ ההשמעות שלנו). בכל צומת ברשימת ההשמעות נעשה מטענת העזר (מספר הצמתים בעץ הפונטרים) O ולכן סך הכל נעשה (סך הפויטרים)O וזה לכל היותר O(n+m).*