***נתאר את מבני הנתונים בהם בחרנו להשתמש עבור התרגיל:***

*מחלקות משמעותיות שמימשנו:*

* ***AvlTree***
  + *עץ AVL דרגות גנרי אשר מאפשר את הפעולות כפי שלמדנו בכיתה – הוספה, הוצאה וחיפוש של איברים בתוכו. מימשנו את העץ כתבנית, כך שכל צומת ברשימה יכול להכיל איברים מכל טיפוס שהוא, כדי שנוכל ליצור עצים מסוגים שונים (עץ אמנים, עץ מספר השמעות ועץ שירים, וכדומה).*
    - *נציין כי מימשנו את העץ כעץ דרגות, ולשם כך נחזיק את גודל תת העץ שהצומת משמש כשורש שלו, גודל תת העץ של הבן השמאלי שלו וגודל תת העץ של הבן הימני שלו (אם אין כאלה – אז נחזיק 0).*
    - *בנוסף, העץ מחזיק בכל רגע את הצמתים המינימלי והמקסימלי תחתיו (לכן נוכל להגיע אליהם ב-).*
* ***HashTable***
  + *טבלת ערבול אשר מאפשרת את הפעולות כפי שלמדנו בכיתה – הוספה, הוצאה וחיפוש של איברים בתוכה. נשתמש בטבלת הערבול על מנת לשמור את כלל האמנים אשר נמצאים כרגע במערכת.*
  + *טבלת הערבול נותנת פתרון להתנגשויות בשיטת Chain-Hashing כפי שנלמד בכיתה (תוך שימוש ב-List אשר מימשנו במבנה הנתונים), ופונקציית הערבול שלה הינה (כאשר x מייצג את ה-artistID, ו-m מייצג את גודל הטבלה, כלומר מספר האיברים במערך).*
    - *בדומה למערך דינמי עליו למדנו בכיתה, טבלת הערבול תשנה את גודלה כשנדרש כך שפקטור העומס, , יקיים בכל רגע במערכת: .* נתאר איך נוודא זאת בהמשך*.*
    - *למדנו בהרצאה כי בשימוש בפונקציית ערבול מסוג מודולו, רצוי ש-m לא יהיה חזקה של 2 או של 10, אך בפיאצה נאמר כי לצרכי התרגיל ניתן להתייחס לפונקציית כנ"ל כמקיימות את עיקרון הפיזור האחיד הפשוט – על כן לא "טרחנו" לוודא כי בשימוש בפונקציה, m אכן לא יהיה חזקה של 2 או של 10.*
* ***List***
  + *רשימה מקושרת דו-כיוונית אשר מאפשרת את פעולות ברירת המחדל של רשימה – הוספה, הוצאה וחיפוש של איברים בתוכה. רשימה זו למעשה אינה גנרית, אלא מתאימה ספציפית לדרישות התרגיל. השימוש ברשימה נעשה רק בתוך טבלת הערבול – כל איבר בטבלת הערבול מכיל רשימה כזו, כאשר כל רשימה מייצגת את כל האמנים המתאימים לאיבר לפי פונקציית הערבול.*
  + *כל צומת ברשימה מייצג אמן – המפתח יהיה ה-artistID המתאים, והוא יכיל מצביע לאיבר מסוג Artist.*
* ***MusicManager***
  + *מבנה כללי ל"מנהל המוזיקה", דרכו נאפשר את כל הפעולות הרצויות במערכת אשר הוגדרו בתרגיל. מבנה זה מכיל את טבלת האמנים (טבלת ערבול אשר תחזיק את כלל האמנים במערכת, להלן טבלת האמנים), את עץ השירים של המערכת, וכן את מספר השירים ומספר האמנים הכולל במערכת בכל רגע.*

*עץ השירים של המערכת:*

* + - *עץ אשר צמתיו מייצגים את כלל השירים במערכת, מסודרים לפי הסדר המתואר בדרישות התרגיל.*
      * *המפתח של כל צומת בעץ –* ***מחלקת Trio****. (וגם האיבר עצמו, שכן רק המפתח מעניין אותנו). השימוש במחלקה נעשה ע"מ שנוכל לממש בקלות את הפונקציה GetRecommendedSongInPlace. המחלקה למעשה מחזיקה 3 פרמטרים – מספר ההשמעות של שיר מסוים + המזהה שלו + המזהה של האמן תחתיו הוא נמצא. ההשוואה בין שני Trio-ים עונה על הדרישה בתרגיל (שווים אמ"ם יש להם מספר השמעות זהה וגם מזהים זהים; Trio אחד גדול מהשני אם מספר ההשמעות שלו קטן יותר, או שמספר ההשמעות שלהם שווים אך ה-artistID גדול שלו יותר, או שהמספר ההשמעות וה-artistID שווים – ה-songID שלו גדול יותר).*

*במימוש הנ"ל, השיר המושמע ביותר יהיה בצומת השמאלי ביותר של העץ, ונוכל להשתמש באלגוריתם למציאת צומת בעץ בעל אינדקס ספציפי, כפי שראינו בתרגול.*

* ***Artist***
  + *להלן אמן. זהו המבנה המייצג כל אמן שהוכנס למערכת.*
  + *לאמן יש מזהה ייחודי (artistID, לפיו הוא גם יישמר בטבלת האמנים), מספר השירים הכולל שלו, עץ של כלל שיריו (ובנוסף משתנה המכיל את מספר השירים הכולל שלו), ועץ של מספר ההשמעות עבור כל שיר שלו.*

*עץ השירים של האמן:*

* + - *עץ אשר צמתיו מייצגים את השירים שהוכנסו תחת האמן במערכת, כשהמפתח של כל צומת – songID.*

*עץ ההשמעות של האמן:*

* + - *עץ אשר צמתיו מייצגים את מספר ההשמעות עבור כל שיר של האמן.*
      * *המפתח של כל צומת בעץ –* ***מחלקת Pair****. השימוש במחלקה נעשה ע"מ שנוכל למצוא בקלות את השיר המושמע ביותר של האמן ב-. המחלקה למעשה מחזיקה זוג פרמטרים – מספר ההשמעות של שיר מסוים + המזהה שלו. ההשוואה בין שני Pair-ים עונה על הדרישה בתרגיל (שווים אמ"ם יש להם מספר השמעות זהה וגם מזהה זהה, ו-Pair אחד גדול מהשני אם מספר ההשמעות שלו גדול יותר, או שמספר ההשמעות שלהם שווים אך ה-songID שלו קטן יותר).*
      * *האיבר בתוך כל צומת בעץ – songID.*

*נתאר מקרה לדוגמה במערכת, אשר ילווה בציור:*

*נניח שבמערכת 8 אמנים, גודל טבלת האמנים 8, ובמערכת 5 שירים באופן כללי;*

*נניח שלאמן 39 יש 4 שירים, והשיר עם מספר ההשמעות הגבוה ביותר הוא שיר 523, אשר הושמע 22 פעמים;*

*נניח גם ש-22 הוא מספר ההשמעות הגבוה ביותר במערכת עבור שיר מסוים, כלומר זהו השיר המושמע ביותר במערכת.*

*במימוש שלנו, המערכת תיראה כך:*

* *קיים MusicManager, המתאר את כלל המערכת שלנו (וכפי שציינו, מאפשר לבצע את הפעולות הנדרשות בתרגיל).*
  + *בטבלת האמנים שלו יש 8 צמתים בכלל הרשימות, כאשר אחד מהם הוא מצביע לאמן עבורו artist\_id = 39.*
    - *לאמן מס' 39 יש עץ שירים בגודל 4, וכן עץ השמעות בגדול 4.*
      * *האינדקס 523 (אשר מייצג את שיר מס' 523) הוא האינדקס השני הכי גבוה בעץ השירים של האמן.*
      * *שיר מס' 523 שהושמע 22 פעמים הוא השיר המושמע ביותר של האמן.*
  + *בעץ השירים של המערכת הצומת הראשון (השמאלי ביותר) מייצג את שיר מס' 523 של אמן מס' 39.*

*להלן איור הממחיש את מבנה המערכת*

*(האיור מכיל נתונים המתאימים עבור המקרה לדוגמה שתיארנו)*

***MusicManager***

*table\_size = 8*

*total\_num\_of\_songs*

*total\_num\_of\_artists*

*8*

*5*

*0*

*1*

*2*

*3*

*4*

*5*

*6*

*7*

***39***

*Trio*

***{ 22 , 39 , 523 }***

*עץ השירים (של המערכת)*

***Artist***

*artist\_id = 39*

*songs\_tree*

*streams\_tree*

*total\_num\_of\_songs = 4*

***523***

*עץ השירים (של האמן)*

*Pair*

***{ 22, 523 }***

*עץ ההשמעות (של האמן)*

***מימוש כל אחת מהפעולות הנדרשות***

*ראשית, נציג מדוע פקטור העומס של טבלת הערבול תמיד .*

*נגדיר: מס' האמנים בטבלה = n, גודל הטבלה = m. נשנה את גודל המערך בשני מקרים:*

1. *: נגדיל את גודל הטבלה פי 2, בכך שניצור טבלה חדשה גדולה פי 2 (ב-), ונעתיק את הטבלה הישנה אליה (נעבור על כל איבר בטבלה ונעתיק כל צומת ברשימה כלשהי המייצג אמן; העתקת אמן לוקחת שכן מדובר בהעתקת משתנים ומצביעים, ונעבור על m צמתים , לכן הדבר לוקח ).*
2. *: באופן דומה, נקטין את גודל הטבלה פי 2, בכך שניצור טבלה חדשה קטנה פי 2 (ב-), ונעתיק את הטבלה הישנה אליה (נעבור על כל איבר בטבלה ונעתיק כל צומת ברשימה כלשהי המייצג אמן; העתקת אמן לוקחת שכן מדובר בהעתקת משתנים ומצביעים, ונעבור על צמתים , לכן הדבר לוקח ).*

*נבחין כי בשיטה זו לשינוי גודל הטבלה, נדאג שמתקיים כי , לכן תמיד , כלומר .*

*למעשה, הדבר מאפשר לנו הכנסה, הוצאה וחיפוש של איבר ב- בממוצע על הקלט משוערך.*

*הראינו ש- תמיד יישאר , לכן פעולת הכנסה/הוצאה/חיפוש לוקחת בממוצע על הקלט, כפי שראינו בכיתה.*

*בנוסף, ראינו בתרגול כי הסיבוכיות של סדרה של m פעולות הכנסה/הוצאה למערך דיני היא , ומכיוון שכל פעולה כזו קורית ב- בממוצע נקבל בסך הכל כי סדרת הפעולות לוקחת בממוצע על הקלט, לכן הסיבוכיות של כל פעולה כהזו היא בממוצע על הקלט משוערך.*

***Init:***

*בעת איתחול מבנה הנתונים, אנו מקצים זיכרון עבור MusicManager, ומשתמשים בבנאי שלו ע"מ ליצור-*

* *טבלת אמנים ריקה. אנו למעשה יוצרים טבלת ערבול בגודל 1, כלומר מערך בגודל 1 (יצירתו קורית ב-), ובאיבר שלו אנו יוצרים רשימה ריקה – גם ב-. אתחול המשתנים בטבלה לוקח גם .*
* *עץ שירים ריק. יצירת העץ קורית ב-, שכן אנו מאתחלים בסה"כ מספר קבוע של משתנים עבור מבנה העץ.*
* *בנוסף, אנו מתאחלים 2 משתנים פרימיטיביים עבור ה-MusicManager, כמובן גם ב-.*

*מכיוון שאנחנו עושים לאורך כל תהליך האיתחול פעולות – אנו עונים על דרישות הסיבוכיות.*

*\* שגיאות: (אם לא היו כאלה, הפונקציה תחזיר מצביע למבנה הנתונים שלנו)*

* *אם לא הצלחנו ליצור MusicManager, הפונקציה תחזיר NULL.*

***AddArtist:***

*בעת הוספת אמן חדש למערכת, נשתמש בבנאי של Artist ליצירת האמן ונוסיפו לטבלת האמנים של MusicManager. בעת יצירת האמן, אנו מאתחלים את המזהה שלו ואת מספר השירים הכולל ב-. בנוסף, אנו יוצרים שני עצים חדשים עבורו – עץ שירי האמן ועץ השמעות האמן. את שני העצים אנו יוצרים ב-, שכן אנו מאתחלים בסה"כ מספר קבוע של משתנים עבור שני עצים ריקים (שכן המערכת לא מאפשרת להוסיף אמן עם שירים בפעולה אחת).*

*בעת הוספת האמן לטבלת האמנים, אנו משתמשים באלגוריתם להכנסת איבר לטבלת הערבול כפי שלמדנו – נשתמש בפונקציית הערבול שהגדרנו ע"מ למצוא את האינדקס של האיבר בטבלה אליו נרצה להכניס את האמן (למעשה, למצוא לאיזו רשימה נוסיף את האמן).*

*לבסוף, נגדיל את המשתנה המחזיק את מספר האמנים בטבלת הערבול ב-1.*

*\* אם לאחר הוספת האמן נגיע למצב בו מתקיים , נגדיל את הטבלה פי 2, וכפי שתיארנו ב- – הדבר לוקח בממוצע על הקלט משוערך.*

***לכן, בסופו של דבר הפעולה תיקח בממוצע על הקלט משוערך כנדרש.***

*\* שגיאות: (אם לא היו כאלה, הפונקציה תחזיר SUCCESS)*

* *אם הייתה בעיה בהקצאת זיכרון, הפונקציה תחזיר ALLOCATION\_ERROR.*
* *אם קיים אמן עם המזהה (נבדוק ע"י חיפוש של בממוצע בטבלת האמנים) – הפונקציה תחזיר FAILURE.*
* *אם DS או artistID לא תקינים לפי הדרישות (נוכל לבדוק זאת ב-) – הפונקציה תחזיר INVALID\_INPUT.*

***RemoveArtist:***

*בעת מחיקת אמן קיים במערכת, אנו מוצאים תחילה את האמן בטבלת האמנים של MusicManager ע"י חיפוש בטבלת הערבול, שכפי שראינו בכיתה לוקח בממוצע על הקלט.*

*כעת, אנו מוחקים את התוכן אליו מצביע הצומת של האמן, כלומר נמחק את ה-Artist המתאים, ב- (שכן המערכת מאפשרת מחיקה של אמן ללא שירים בלבד, ולכן המחיקה תהיה של משתנים ושל עצים ריקים). לאחר מכן, אנו מוחקים את הצומת עצמו ב- (שכן מדובר רק במחיקה והחלפה של מספר קבוע של מצביעים ברשימה).*

*לבסוף, נקטין את המשתנה המחזיק את מספר האמנים בטבלת הערבול ב-1.*

*\* אם לאחר הוספת האמן נגיע למצב בו מתקיים , נקטין את הטבלה פי 2, וכפי שתיארנו ב- – הדבר לוקח בממוצע על הקלט משוערך.*

***לכן, בסופו של דבר הפעולה תיקח בממוצע על הקלט משוערך כנדרש.***

*\* שגיאות: (אם לא היו כאלה, הפונקציה תחזיר SUCCESS)*

* *אם הייתה בעיה בהקצאת זיכרון, הפונקציה תחזיר ALLOCATION\_ERROR.*
* *אם לא קיים אמן עם המזהה (נבדוק בחיפוש של בממוצע בטבלת האמנים) – הפונקציה תחזיר FAILURE.*
* *אם DS או artistID לא תקינים לפי הדרישות (נוכל לבדוק זאת ב-) – הפונקציה תחזיר INVALID\_INPUT.*

***AddSong:***

*בעת הוספת שיר חדש לאמן מסוים במערכת, אנו מוצאים תחילה את האמן בטבלת האמנים של MusicManager ע"י חיפוש בטבלת ערבול, שכפי שראינו בכיתה לוקח בממוצע על הקלט.*

*משיש בידינו את האמן, אנו מוסיפים צומת חדש תחת עץ השירים של האמן (יצירת הצומת לוקחת שכן הוא מחזיק מספר קבוע של משתנים, והוספתו לעץ לוקחת אם הוא מספר השירים של האמן, כפי שלמדנו שלוקחת הוספה לעץ AVL), וכן מוסיפים צומת חדש מתאים גם תחת עץ ההשמעות של האמן (ומאותן סיבות, גם פעולה זו לוקחת ). לבסוף, אנו מגדילים את המשתנה המחזיק את מספר השירים תחת האמן ב-1.*

*לאחר מכן, אנו עוברים להוסיף את השיר בעץ השירים של המערכת. לשם כך, אנו יוצרים צומת חדש אשר ייצג את השיר (גם במקרה הנ"ל, ב-) ומוסיפים אותו לעץ השירים הכולל ב- (כאשר הוא מספר השירים הכולל במערכת, כפי שלמדנו שלוקחת הוספה לעץ AVL). לבסוף, אנו מגדילים את מספר השירים הכולל במערכת ב-1.*

*נוכל לראות כי (הרי שלא ייתכן כי לאמן מסוים יש מספר גדול יותר של שירים מבכל המערכת), ולכן גם . אזי, נשים לב כי עשינו פעולות ע"מ להוסיף את השיר לאמן, פעולות ע"מ להוסיף את השיר לעץ השירים של המערכת, ו- בממוצע על הקלט ע"מ למצוא את האמן בטבלת האמנים.*

***לכן, בסופו של דבר הפעולה תיקח בממוצע על הקלט כנדרש.***

*\* שגיאות: (אם לא היו כאלה, הפונקציה תחזיר SUCCESS)*

* *אם הייתה בעיה בהקצאת זיכרון, הפונקציה תחזיר ALLOCATION\_ERROR.*
* *אם לא קיים אמן עם המזהה (נבדוק בחיפוש של בממוצע בטבלת האמנים) – הפונקציה תחזיר FAILURE.*
* *אם קיים שיר עם המזהה תחת האמן הנ"ל (נבדוק ע"י חיפוש של בעץ ההשמעות תחת האמן) – הפונקציה תחזיר FAILURE.*
* *אם songID, DS או artistID לא תקינים לפי הדרישות (נוכל לבדוק זאת ב-) – הפונקציה תחזיר INVALID\_INPUT.*

***RemoveSong:***

*בעת מחיקת שיר קיים מאמן מסוים במערכת, אנו מוצאים תחילה את האמן בטבלת האמנים של MusicManager ע"י חיפוש בטבלת ערבול, שכפי שראינו בכיתה לוקח בממוצע על הקלט.*

*משיש בידינו את האמן, אנו יוצרים משתנה חדש אשר מחזיק את מספר ההשמעות של השיר הנ"ל (יצירת המשתנה לוקחת , ומציאת מספר ההשמעות של השיר מתוך עץ ההשמעות של האמן לוקחת אם הוא מספר השירים של האמן, הרי שכך לוקח חיפוש בעץ AVL כפי שלמדנו), ומוחקים את השיר מעץ השירים וגם מעץ ההשמעות של האמן (מחיקת השיר תחת כל עץ כנ"ל לוקחת גם היא, כפי שלמדנו שלוקחת מחיקה מעץ AVL, ומחיקת השיר עצמו לוקחת , שכן השיר מחזיק מספר קבוע של משתנים). לבסוף, אנו מקטינים את המשתנה המחזיק את מספר השירים תחת האמן ב-1.*

*לאחר מכן, אנו עוברים למחוק את השיר מעץ השירים של המערכת. לשם כך, אנו נעזרים במשתנה אשר מחזיק את מספר ההשמעות של השיר הנ"ל ע"מ למצוא את הצומת המתאים בעץ השירים של המערכת, ומוחקים את הצומת מהעץ (ב-, כאשר n הוא מספר השירים הכולל במערכת, כפי שלמדנו שלוקחת מחיקה מעץ AVL)*

*לבסוף, אנו מקטינים את מספר השירים הכולל במערכת ב-1.*

*שוב נוכל לראות כי , ולכן גם . אזי, נשים לב כי עשינו פעולות ע"מ למחוק את השיר תחת האמן, פעולות ע"מ למחוק את השיר מעץ השירים של המערכת, ו- בממוצע על הקלט ע"מ למצוא את האמן בטבלת האמנים.*

***לכן, בסופו של דבר הפעולה תיקח בממוצע על הקלט כנדרש.***

*\* שגיאות: (אם לא היו כאלה, הפונקציה תחזיר SUCCESS)*

* *אם הייתה בעיה בהקצאת זיכרון, הפונקציה תחזיר ALLOCATION\_ERROR.*
* *אם לא קיים אמן עם המזהה (נבדוק בחיפוש של בממוצע בטבלת האמנים) – הפונקציה תחזיר FAILURE.*
* *אם לא קיים שיר עם המזהה תחת האמן הנ"ל (נבדוק ע"י חיפוש של בעץ ההשמעות תחת האמן) – הפונקציה תחזיר FAILURE.*
* *אם songID, DS או artistID לא תקינים לפי הדרישות (נוכל לבדוק זאת ב-) – הפונקציה תחזיר INVALID\_INPUT.*

***AddSongToCount:***

*בעת הוספת כמות השמעות לשיר קיים תחת אמן מסוים במערכת, אנו מוצאים תחילה את האמן בטבלת האמנים של MusicManager ע"י חיפוש בטבלת ערבול, שכפי שראינו בכיתה לוקח בממוצע על הקלט.*

*משיש בידינו את האמן, אנו מחפשים את השיר הספציפי בעץ ההשמעות תחת האמן ומוסיפים לו את כמות ההשמעות שקיבלנו כפרמטר לפונקציה (חיפוש השיר בעץ ההשמעות לוקח אם הוא מספר השירים של האמן, כפי שלמדנו שלוקח חיפוש לעץ AVL, והוספת הכמות לוקחת ). תוך כדי פעולה זו, אנו יוצרים משתנה זמני אשר יחזיק את מספר ההשמעות של השיר לפני שהוספנו לו את כמות ההשמעות הנוספת, ע"מ שנוכל למצוא את השיר שוב בעץ השירים של המערכת.*

*לאחר מכן, ניעזר במשתנה הזמני הנ"ל (ובמזהה השיר והאמן כמובן) ע"מ למצוא את הצומת אשר מייצג את השיר עם כמות ההשמעות ה"ישנה" (כלומר זו לפני ההוספה בפונקציה זו) בעץ השירים הכולל של המערכת, ולמחוק אותו מהעץ (ב- כאשר n הוא מספר השירים הכולל במערכת, כפי שלמדנו שלוקחת מחיקה מעץ AVL). כעת, ניצור צומת חדש ונוסיף אותו לעץ השירים של המערכת (שוב ב-, כפי שלמדנו שלוקחת הוספה לעץ AVL).*

*גם במקרה הנ"ל, נוכל לראות כי , ולכן גם . אזי, נשים לב כי עשינו פעולות ע"מ לשנות את כמות ההשמעות של השיר תחת האמן, פעולות ע"מ לשנות את כמות ההשמעות של השיר תחת עץ השירים של המערכת, ו- בממוצע על הקלט ע"מ למצוא את האמן בטבלת האמנים.*

***לכן, בסופו של דבר הפעולה תיקח בממוצע על הקלט כנדרש.***

*\* שגיאות: (אם לא היו כאלה, הפונקציה תחזיר SUCCESS)*

* *אם הייתה בעיה בהקצאת זיכרון, הפונקציה תחזיר ALLOCATION\_ERROR.*
* *אם לא קיים אמן עם המזהה (נבדוק בחיפוש של בממוצע בטבלת האמנים) – הפונקציה תחזיר FAILURE.*
* *אם לא קיים שיר עם המזהה תחת האמן הנ"ל (נבדוק ע"י חיפוש של בעץ ההשמעות תחת האמן) – הפונקציה תחזיר FAILURE.*
* *אם count, songID, DS או artistID לא תקינים לפי הדרישות (נוכל לבדוק זאת ב-) – הפונקציה תחזיר INVALID\_INPUT.*

***GetArtistBestSong:***

*בעת חיפוש השיר המושמע ביותר של אמן מסוים, אנו מוצאים תחילה את האמן בטבלת האמנים של MusicManager ע"י חיפוש בטבלת ערבול, שכפי שראינו בכיתה לוקח בממוצע על הקלט.*

*מכיוון שאנו שומרים במבנה הנתונים שלנו עץ השמעות עבור כל אמן, ולכל עץ כזה מחזיקים מצביע לצומת הגדול ביותר בו ("גדול" לפי ההגדרות בתרגיל), נוכל ב- להגיע לשיר המושמע ביותר של האמן (הרי שנוכל לקבל את ה-songID של הצומת המקסימלית של עץ ההשמעות תחת האמן הרצוי).*

***לכן, בסופו של דבר הפעולה לוקחת בממוצע על הקלט כנדרש.***

*\* שגיאות: (אם לא היו כאלה, הפונקציה תחזיר SUCCESS)*

* *אם הייתה בעיה בהקצאת זיכרון, הפונקציה תחזיר ALLOCATION\_ERROR.*
* *אם לא קיים אמן עם המזהה (נבדוק בחיפוש של בממוצע בטבלת האמנים) – הפונקציה תחזיר FAILURE.*
* *אם לא קיים שיר עם המזהה תחת האמן הנ"ל (נבדוק ע"י חיפוש של בעץ ההשמעות תחת האמן) – הפונקציה תחזיר FAILURE.*
* *אם songID, DS או artistID לא תקינים לפי הדרישות (נוכל לבדוק זאת ב-) – הפונקציה תחזיר INVALID\_INPUT.*

***GetRecommendedSongInPlace:***

*בעת החזרת השיר במקום rank במצעד הלהיטים, אנו תחילה מגיעים לעץ השירים של המערכת (ב-, שהרי ישנו מצביע לעץ הנ"ל תחת MusicManager).*

*כעת, נשתמש באלגוריתם למציאת הצומת בעץ בעל האינדקס rank – אותו אלגוריתם אשר ראינו בתרגול. כפי שלמדנו, האלגוריתם לוקח , כאשר n הוא מספר השירים הכולל במערכת (ולכן גם מספר האיברים בעץ הנ"ל). במפתח של צומת הנ"ל אנו מחזיקים ב-Trio, אשר כפי שתיארנו קודם לכן מורכב ממספר ההשמעות של השיר + מזהה השיר+ מזהה האמן, ולכן נוכל ב- להחזיר את 3 הפרמטרים הנ"ל של השיר מהפונקציה (כלומר לשנות את הערך אליהם מצביעים המצביעים שקיבלנו כפרמטרים לפונקציה)..*

***לכן, בסופו של דבר הפעולה לוקחת במקרה הגרוע כנדרש (מטענות שראינו בכיתה).***

*\* שגיאות: (אם לא היו כאלה, הפונקציה תחזיר SUCCESS)*

* *אם הייתה בעיה בהקצאת זיכרון, הפונקציה תחזיר ALLOCATION\_ERROR.*
* *אם (נוכל לבדוק זאת ב-) – הפונקציה תחזיר FAILURE.*
* *אם rank, songID, DS או artistID לא תקינים לפי הדרישות (נוכל לבדוק זאת ב-) – הפונקציה תחזיר INVALID\_INPUT.*

***Quit:***

*הפונקציה, כפי שהיא ממומשת במבנה הנתונים שלנו, קוראת למעשה להורס של ה-MusicManager.*

*ההורס הנ"ל עובר תחילה על כל הצמתים המייצגים אמנים בטבלת האמנים (ב-, כש-m הינו מספר האמנים בטבלה\*), ולכל צומת מוחק את התוכן אליו הוא מצביע – הוא למעשה ה-Artist המתאים (אם נסמן ב- את מספר השירים של האמן הנ"ל, אזי לאמן יש 2 עצים בגודל ועוד מספר קבוע של משתנים, לכן מחיקתו לוקחת ).*

*נוכל לשים לב כי מחיקת כל האמנים לוקחת עבור המעבר על כל האמנים, בנוסף ל- לכל אמן. סך השירים, כלומר סכימת ה--ים של כל אמן בטבלת האמנים, מסתכם ל- – פעמיים מספר השירים הכולל במערכת. אזי, מחיקת כל האמנים לוקחת (מטענות שראינו בכיתה).*

*\* חשוב לציין כי לפי , בכל רגע נתון במערכת מתקיים , לכן אין תוספת שצריך לקחת בחשבון בעקבות מעבר על איברים ריקים בטבלה.*

*לאחר מכן, ההורס עובר שוב על כל הצמתים בטבלה, והפעם מוחק את הצמתים עצמם (שוב ב-), את הרשימות הריקות שהן למעשה האיברים של הטבלה (ב-, שכן כל רשימה היא ריקה בעת מחיקתה), ולאחר מכן את הטבלה כולה (ב- כמובן).*

***לכן, בסופו של דבר הפעולה לוקחת כנדרש (מטענות שראינו בכיתה).***

*\* שגיאות: לא אמורות להיות.*

***סיבוכיות מקום:***

*נתבונן בזיכרון שתופס כל מבנה נתונים במערכת שלנו, כלומר ב-****MusicManager****:*

* ***טבלת האמנים****: ישנם m צמתים של אמנים בטבלת האמנים\*, ולכל אמן יש שני עצים המכילים צמתים (כאשר מציין את מספר השירים של האמן הנ"ל). לכן, סך השירים, כלומר סכימת ה--ים של כל אמן בטבלת האמנים, מסתכם ל- – פעמיים מספר השירים הכולל במערכת.*

*בתוספת המספר הקבוע של מקום שנשמר יחד עם כל שיר/אמן/בטבלה עצמה, נקבל כי סיבוכיות המקום של טבלת האמנים היא .*

*\* חשוב לציין כי לפי , בכל רגע נתון במערכת מתקיים , לכן אין צורך להתייחס בחישוב סיבוכיות המקום שלנו לאיברים הריקים הנוספים בטבלה.*

* ***עץ השירים של המערכת****: בעץ הנ"ל ישנם n צמתים (שכן הוא מייצג את כלל השירים הקיימים במערכת), כאשר כל צומת שכזה מכיל בתוכו אך ורק מספר קבוע של משתנים.*

*אם כך, סיבוכיות המקום של עץ השירים של המערכת היא .*

***לכן, נקבל כי סיבוכיות המקום הכוללת היא (מטענות שראינו בכיתה).***