נתאר את מבני הנתונים בהם בחרנו להשתמש עבור התרגיל:

<u>מחלקות משמעותיות שמימשנו</u>:

- ***AvlTree***
  - עץ AVL 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 כך שבכל רגע נתון נחזיק את הצומת המינימלי בו (לכן נוכל להגיע אליה ב-$O(1)$).
- ***StreamList***
  - 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.
  - כל צומת ברשימה מייצג את מספר ההשמעות (הצומת הראשון – "0" השמעות, הצומת האחרון – מקס' ההשמעות לשיר במערכת ברגע נתון), וכן מכיל בתוכו איבר מטיפוס עץ של מצביעים לעץ השמעות.
- ***MusicManager***
  - מבנה כללי ל"מנהל המוזיקה", דרכו נאפשר את כל הפעולות הרצויות במערכת אשר הוגדרו בתרגיל. מבנה זה מכיל עץ אמנים (כלל האמנים אשר הוכנסו למערכת, להלן עץ האמנים), רשימה מסוג *StreamList* (להלן רשימת ההשמעות), וכן את מספר השירים הכולל במערכת.
- ***Artist***
  - להלן אמן. זהו מבנה המייצג כל אמן שהוכנס למערכת.
  - לאמן יש מזהה ייחודי (*artist_id*, לפיו הוא גם יישמר בעץ האמנים), מערך של כלל שיריו (ובנוסף משתנה המכיל את מספר השירים הכולל), וכן הוא מכיל עץ של מספר השמעות.

    <u>מערך שירים</u>:
    - מערך בגודל מס' השירים הכולל של האמן (*num_of_songs*). כל איבר במערך יהיה מצביע לצומת ברשימה *StreamList* אשר מייצג מספר ההשמעות שהיו לשיר הנ"ל במערכת.

    <u>עץ של מספר השמעות</u>:
    - עץ של מספר השמעות יכיל את כל מספרי ההשמעות שיש לאמן עבור כל שיריו (למשל, אם לאמן 2 שירים עם 0 השמעות – בעץ זה יהיה צומת אחד המייצג "0" השמעות). כל צומת יסודר בעץ ע"פ מספר ההשמעות של השירים תחתיו.
    - בכל צומת של העץ הנ"ל, יהיה איבר של עץ שירים – עץ המייצג את כל השירים של האמן אשר הושמעו X פעמים (למשל, בדוגמה הקודמת, תחת הצומת של "05" השמעות שבעץ מספר ההשמעות – יהיו שני צמתים המייצגים את שני השירים של האמן, מסודרים לפי *song_id*).

נתאר מקרה לדוגמה במערכת, אשר ילווה בציור:

נניח שבמערכת 5 אמנים;

לאמן 17 יש 4 שירים, והשיר עם מספר ההשמעות הגבוה ביותר הוא שיר 1, אשר הושמע 22 פעמים;

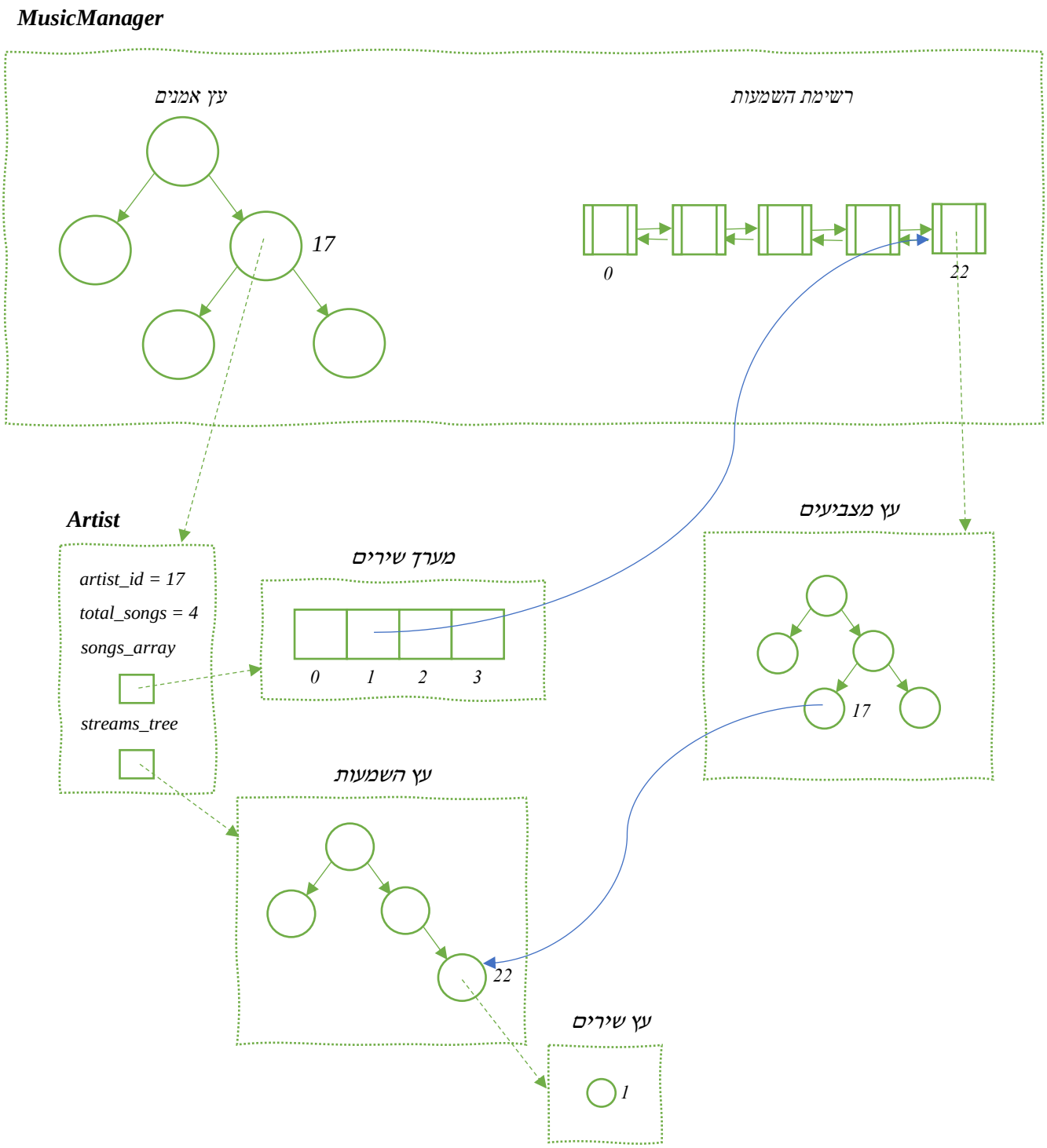
נניח גם ש-22 הוא מספר ההשמעות הגבוה ביותר במערכת עבור שיר מסוים.

במימוש שלנו, המערכת תיראה כך:

- קיים *MusicManager*, אשר למעשה מתאר את כלל המערכת שלנו (וכפי שציינו, מאפשר לבצע את כלל הפעולות שנדרשו בתרגיל).
  - בעץ האמנים שלו יש 5 צמתים, כאשר אחד מהם הוא האמן עבורו *artist_id = 17*.
    - לאמן מס' 17 יש מערך שירים בגודל 4, וכן עץ השמעות עם מס' צמתים לפי כמות השירים שהושמעו מספר שונה של פעמים עבור האמן הנ"ל.
      - האינדקס 1 (אשר מייצג את שיר מס' 1) מכיל מצביע לצומת אשר מייצג "22" השמעות ברשימת ההשמעות.
    - תחת הצומת 22 נמצא עץ השירים של האמן הנ"ל אשר הושמעו 22 פעמים – במקרה של הדוגמה שלנו יהיה זה שיר מס' 1.
  - ברשימת ההשמעות שלו יש מספר מסוים של צמתים (לפי כמות השירים שהושמעו מספר שונה של פעמים בכלל המערכת), כאשר הצומת הראשון מייצג "0" השמעות, והצומת האחרון מייצג את מספר ההשמעות הגבוה ביותר במערכת עם שיר כלשהו – "22" השמעות במקרה של הדוגמה שלנו.
    - הצומת "22" מכיל עץ מסוג מצביע לעץ השמעות, ובין השאר מכיל את הצומת "17" אשר מייצגת את אמן מס 17, ומכילה מצביע לצומת "22" בעץ ההשמעות של אמן זה.

<u>להלן איור הממחיש את מבנה המערכת</u>

(האיור מכיל נתונים המתאימים עבור המקרה לדוגמה שתיארנו)



<u>מימוש כל אחת מהפעולות הנדרשות</u>

נציג טענות אשר יעזרו לנו בהוכחות הסיבוכיות, ואז נעבור להסברים על מימוש הפעולות ווהוכחת הסיבוכיות הנדרשת:

(*)**ניתן לעבור על r צמתים בעץ כלשהו בסיבוכיות $O(r)$, במקרה בו יש בידינו את הצומת המינימלי בעץ:**

נתחיל בצומת המינימלי בעץ, ונסייר *in-order*. הצומת נמצא בעלה הכי שמאלי של העץ, וכדי להגיע לצומת ה-r נעשה (מספר העליות שעשינו) O. מכיוון שאנו מסיירים *in-order*, נבצע את הפעולה שאנו רוצים לעשות על כל צומת בעת העלייה מהבן השמאלי, ולא נבצעה בעלייה מהבן הימני. בכל פעם שנעבור מצומת מסוים לצומת העוקב לו, נבצע פעולות ב-(מספר הפעולות שכבר ביצענו) O, מכיוון שאם נרד ממנו לצומת המינימלי זה ייקח לנו לכל היותר את מספר הפעולות שלקח לנו להגיע מהצומת השמאלי התחתון ביותר (הצומת המינימלי) אל הצומת המסוים שאנו נמצאים בו.

מכאן, שעבור כל r, מספר העליות שביצענו הוא $O(r)$, ולכן סך הכל הפעולות שנעשה הוא $O(r)$, כפי שרצינו להראות.

(**) **כמות השירים שהושמעו מספר שונה של פעמים בכלל המערכת (או עבור כל אמן) בהכרח קטנה או שווה למספר השירים הכולל במערכת (או עבור כל אמן):** נניח לאמן כלשהו יש X שירים. נניח בשלילה כי יש בעץ ההשמעות שלו $X+1$ צמתים ומעלה. מכיוון שלא ייתכן כי לשיר אחד יש 2 מספרי השמעות (שיר בהכרח הושמע מספר מדויק של פעמים בכל רגע נתון), נקבל כי בהכרח צומת מסוים בעץ ההשמעות יהיה ריק – וזה בלתי אפשרי, אחרת הצומת לא היה קיים (שכן אין לאמן שירים שהושמעו מספר כזה של פעמים). קיבלנו סתירה, ולכן הוכחנו את הטענה.

(***) **מספר האמנים הכולל במערכת בהכרח קטן או שווה למספר השירים במערכת:** המערכת לא מאפשר להכניס אליה שיר שאינו משויך לאמן, ולכן לכל אמן יש לכל הפחות שיר אחד.

(****) **ניתן לבנות עץ בעל n צמתים ממערך ממויין בגודל n המכיל את המספרים *0,1,2,…,n-1* ב-$O(n)$:**

האלגוריתם בו השתמשנו במימוש שלנו נראה כך – נעבור על המערך באופן רקורסיבי כך שבכל פעם נחלק את המערך לחצי עליון וחצי תחתון (לפי גודל המספרים שבו). לאחר שהגענו לתנאי העצירה (בו אורך המערך הוא 0), נכניס לשורש את האיבר באמצע המערך, לבן השמאלי את חצי המערך הקטן יותר ולבן הימני את זה הגדול יותר (נכניס *NULL* במקומות בהם המערך הוא 0). כך בכל פעם שאנו חוזרים מהרקורסיה הפנימית אנו יוצרים עץ שהוא כבר עץ *AVL* תקין, עד שלבסוף נקבל עץ עם כל האיברים במערך. במהלך האלגוריתם אנו יוצרים n צמתים, ולכן אנו עושים $O(n)$ פעולות.

***<u>Init</u>*:**

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

- עץ ריק של אמנים. יצירת העץ קורית ב-$O(1)$, כיוון שאנו מאתחלים בסך הכל מספר קבוע של משתנים עבור המבנה של העץ.
- רשימה מקושרת של מספר ההשמעות, ובה יש צומת אחד המייצגת "0" השמעות. בצומת הנ"ל נוצר עץ ריק של מצביעים לאמנים. גם יצירה זו קורית ב-$O(1)$ (יצירת העץ, יצירת צומת אחד ויצירת רשימה אחת – כולם מאתחלים בסך הכל מספר קבוע של משתנים עבור המבנה).

מכיוון שאנחנו עושים לאורך כל תהליך האיתחול $O(1)$ פעולות – אנו עונים על דרישות הסיבוכיות.

* <u>שגיאות</u>: (אם לא היו כאלה, הפונקציה תחזיר מצביע למבנה הנתונים שלנו)

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

<u>*AddArtist*</u>:

בעת הוספת אמן חדש למערכת, נשתמש בבנאי של *Artist* ליצור את האמן ולהוסיפו לעץ האמנים של *MusicManager*.
ביצירת האמן, אנו מאתחלים את המזהה שלו ואת מספר השירים הכולל ב-$O(1)$. בנוסף, אנו יוצרים מערך חדש בגודל m, ומאתחלים כל איבר בו להצביע לצומת "0" ברשימת ההשמעות – פעולה שלוקחת $O(m)$. לבסוף, אנו יוצרים עץ השמעות חדש עם צומת אחד של "0" השמעות (לוקח $O(1)$), עבורו אנו גם יוצרים עץ שירים חדש שמכיל צמתים עבור כל השירים של האמן, ולכן יש לו m צמתים. את יצירת עץ השירים הנ"ל אנו עושים ב-$O(m)$ – ע"י האלגוריתם אשר מוסבר בטענה (****). אזי, יצירת האמן לוקחת $2 \cdot O(m)$, כלומר $O(m)$ לפי מה שלמדנו בכיתה.
בהוספת האמן שיצרנו ל-*MusicManager*, ייקח לנו $O(\log n)$ להוסיף אותו במקום הנכון (כפי שלמדנו בכיתה על הוספת איבר לעץ *AVL*), ועוד $O(1)$ להוסיף ל-*MusicManager* את מס' השירים שנוספו למערכת.
לכן, בסופו של דבר נבצע $O(m) + O(\log n) = O(m + \log n)$ פעולות כנדרש (מטענה שראינו בכיתה).

* <u>שגיאות</u>: (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 $O(\log n)$ בעץ האמנים) – הפונקציה תחזיר *FAILURE*.
- אם *numOfSongs*, *DS* או *artistID* לא תקינים לפי הדרישות (נוכל לבדוק זאת ב-$O(1)$) – הפונקציה תחזיר *INVALID_INPUT*.

<u>*RemoveArtist*</u>:

בעת מחיקת אמן קיים מהמערכת, אנו מוצאים תחילה את האמן בעץ האמנים של *MusicManager* ע"י חיפוש בעץ *AVL*, שכפי שראינו בכיתה לוקח $O(\log n)$.
משיש בידינו את האמן, אנו רצים בלולאה על מערך השירים הכולל שלו ובעזרתו מוחקים את הצמתים המצביעים על האמן מכלל העצים שבתוך כל צומת שברשימת ההשמעות:

- מתחילים במצביע שבאינדקס 0 במערך השירים הכולל, ומגיעים ב-$O(1)$ לעץ המתאים תחת הצומת X ברשימת ההשמעות, כאשר X מתאר את מספר ההשמעות של השיר מס' 0.
- מוצאים את המצביע לאמן בתוך העץ הנ"ל ב-$O(\log n)$, כפי שתיארנו קודם עבור עץ *AVL*.
- נעזרים במצביע הנ"ל להגיע לעץ השירים שתחת עץ ההשמעות של האמן הספציפי בעץ האמנים של *MusicManager*, וע"י סיור *in-order* (שמתחיל בצומת הקטן ביותר, אליו אנו יכולים ב-$O(1)$) עוברים על כל השירים עם X השמעות ומשנים בתוך מערך השירים את ערך המצביע שבאינדקסים שמצאנו (לפי *song_id*) ל-*NULL* (ראינו בכיתה כי סיור *in-order* שכזה ייקח $O(m_1)$, כאשר m_1 הוא מספר השירים של האמן עם X השמעות, כלומר מספר האיברים בעץ אשר בצומת X).
- כשמסיימים לעבור על כל עץ השירים הרלוונטי, עוברים לאינדקס 1 (האינדקס הבא במערך), ועושים את אותה פעולה. אם המצביע האינדקס הוא *NULL*, אנו עוברים ישר לאינדקס הבא.

- כך למעשה אנו רצים על כל האיברים במערך השירים, כלומר נעשה $O(m) \cdot O(\log n)$ פעולות כדי לעבור על כלל המערך. בכך שאנו משנים את המצביעים ל-*NULL* במהלך הריצה בלולאה, אנו דואגים שבכל הריצות של הלולאה אנו עושים לכל היותר $O(m_1) + O(m_2) + \cdots + O(m_i) = O(m_{tot})$ פעולות, כאשר m_{tot} הוא מספר הצמתים הכולל בעץ ההשמעות, ומטענה $(**)$ נקבל כי זהו מספר שחסום ב-m, ולכן אנו עושים לכל היותר $O(m)$ פעולות נוספות.

בנוסף, לוקח לנו $O(1)$ להוריד ל-*MusicManager* את מס' השירים שנמחקו מהמערכת.

בסופו של דבר, נבצע $O(m) \cdot O(\log n) + O(m) = O(m \log n)$ פעולות כנדרש (מטענה שראינו בכיתה).

* <u>שגיאות</u>: (אם לא היו כאלה, הפונקציה תחזיר *SUCCESS*)

- אם הייתה בעיה בהקצאת זיכרון, הפונקציה תחזיר *ALLOCATION_ERROR*.
- אם לא קיים אמן עם המזהה (נבדוק ע"י חיפוש $O(\log n)$ בעץ האמנים) – הפונקציה תחזיר *FAILURE*.
- אם *DS* או *artistID* לא תקינים לפי הדרישות (נוכל לבדוק זאת ב-$O(1)$) – הפונקציה תחזיר *INVALID_INPUT*.

***<u>AddSongToCount</u>*:**

בעת הוספת השמעה אחת לשיר מס' X של האמן Y, אנו מוצאים תחילה את האמן בעץ האמנים של *MusicManager* ע"י חיפוש בעץ *AVL*, שכפי שראינו בכיתה לוקח $O(\log n)$.

נניח כי לשיר יש כעת M השמעות. משיש בידינו את האמן Y:

- נעדכן את מיקום השיר X תחת עץ ההשמעות של האמן – נמחק את השיר מהצומת "M" בעץ ההשמעות ב-$O(\log m)$ (מחיקת איבר מעץ), ונחפש האם הצומת "$M+1$" קיים בעץ ההשמעות, גם כן ב-$O(\log m)$ (חיפוש איבר בעץ, למעשה ב-$O(\log c)$ כאשר c מסמל את מספר הצמתים בעץ, אבל מטענה $(**)$ נקבל כי גם ב-$O(\log m)$):
  - אם כן, נוסיף את הצומת "X" לעץ השירים של צומת "M" תחת עץ ההשמעות ב-$O(\log m)$ (הוספת איבר לעץ).
  - אם לא, נוסיף את הצומת "$M+1$" לעץ ההשמעות ב-$O(\log m)$ (הוספת איבר לעץ), וניצור לו צומת יחיד עבור השיר "X" (ב-$O(1)$).
- נעדכן את המצביע שבאינדקס X תחת מערך השירים הכולל של אמן Y כך שכבר לא יצביע לצומת "M" השמעות ברשימת ההשמעות, אלא לצומת "$M+1$":
  - אם הצומת "$M+1$" קיים, נוסיף לעץ שהוא מכיל את הצומת "Y" ב-$O(\log n)$ (הוספת איבר לעץ), עם מצביע לצומת "$M+1$" תחת עץ ההשמעות של האמן.
  - אם לא, ניצור ברשימה צומת "$M+1$" ב-$O(1)$ (שכן בסך הכל מדובר בהוספת איבר לרשימה, וראינו בכיתה כי מדובר ב-$O(1)$) אשר יכיל עץ עם איבר יחיד "Y", אשר יצביע לצומת "$M+1$" תחת עץ ההשמעות של האמן.

לאחר מכן, נעדכן את המצביע המתאים במערך השירים של האמן.

- אם השיר X של האמן Y היה השיר היחיד עם M השמעות:
  - נמחק את הצומת "M" מעץ ההשמעות של האמן ב-$O(\log m)$ (מחיקת איבר מעץ).
  - נמחק את הצומת "M" מרשימת ההשמעות ב-$O(1)$ (מחיקת איבר ידוע ברשימה).

נסכום את כלל הפעולות שביצענו במקרה הגרוע, ונקבל כנדרש (מטענות שראינו בכיתה):

$$O(\log n) + O(\log m) + O(\log m) + O(\log m) + O(\log n) + O(\log m) = O(\log n) + O(\log m)$$
$$= O(\log n + \log m)$$

* <u>שגיאות</u>: (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 $O(\log n)$ בעץ האמנים) – הפונקציה תחזיר *FAILURE*.
- אם *numOfSongs*, *songID*, *DS* או *artistID* לא תקינים לפי הדרישות (נוכל לבדוק זאת ב-$O(1)$) – הפונקציה תחזיר *INVALID_INPUT*.

***<u>NumberOfStreams</u>*:**

בעת החזרת מספר ההשמעות של השיר X של האמן Y, אנו מוצאים תחילה את האמן בעץ האמנים של *MusicManager* ע"י חיפוש בעץ *AVL*, שכפי שראינו בכיתה לוקח $O(\log n)$.

כאשר יש בידינו את האמן Y, אנו יכולים לפנות לאינדקס X שבמערך השירים שלו ב-$O(1)$ (המערך נמצא בתוך המחלקה של האמן, וכן פנייה לאינדקס ספציפי במערך לוקחת $O(1)$). בתוך האינדקס הנ"ל ישנו מצביע לצומת ברשימת ההשמעות, צומת אשר מייצג את מספר ההשמעות של השיר X. הצומת הנ"ל מכיל בתוכו (במימוש שלנו) את מספר ההשמעות הרצוי, ולכן נבצע $O(1)$ פעולות על מנת לקבל מספר זה.

מרגע שמספר ההשמעות בידינו, נבצע עוד $O(1)$ פעולות ע"מ לשמור את המספר בכתובת שבמצביע $streams$, ונסיים.

לכן, בסופו של דבר נבצע $O(1) + O(\log n) = O(\log n)$ פעולות כנדרש (מטענה שראינו בכיתה).

* <u>שגיאות</u>: (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 $O(\log n)$ בעץ האמנים) – הפונקציה תחזיר *FAILURE*.
- אם *numOfSongs*, *songID*, *DS* או *artistID* לא תקינים לפי הדרישות (נוכל לבדוק זאת ב-$O(1)$) – הפונקציה תחזיר *INVALID_INPUT*.

***<u>GetRecommendedSongs</u>*:**

בעת החזרת מצעד הלהיטים, נפנה תחילה לצומת האחרון ברשימת ההשמעות של *MusicManager* (אנו מחזיקים ברשימה מצביע לצומת האחרון שלה, לכן זה לוקח $O(1)$), אשר מייצג את מספר ההשמעות הגבוה במערכת – נסמנו M.

הצומת של "M" השמעות מכיל עץ של מצביעים לצמתים "M" בעצי ההשמעות של כל האמנים להם יש לכל הפחות שיר אחד עם M השמעות. אנו ניגשים לצומת המינימלי בעץ המצביעים (נוכל לעשות זאת ב-$O(1)$ שכן אנו מחזיקים מצביע

לצומת המינימלי בעץ), שמחזיק למעשה מצביע לעץ השירים שהושמעו M פעמים של האמן עם ה-*artist_id* המינימלי – נסמן אמן זה ב-$a_{M,1}$.

בעץ השירים שהושמעו M פעמים של $a_{M,1}$, נעבור על השירים ע"י סיור *in-order* שייתן לנו את השירים בסדר עולה לפי *song_id* (ושוב נוכל להתחיל מהאיבר המינימלי, כלומר עם ה-*song_id* המינימלי), כאשר עבור על שיר עליו נעבור נכניס את הערך המתאים למערכי הקלט שקיבלנו כפרמטרים. אנו משתמשים ב-*counter* אשר מתחיל מ-0 וסופר את מספר השירים שכבר הכנסנו למערכים הנ"ל, וכך נוכל לדעת את האינדקס במערכים להכניס אליו את ערכי השיר האמן.

אם מספר השירים שעלינו להכניס למערכים (*numOfSongs*) קטן ממספר השירים שבעץ השירים הנ"ל, נעבור רק על חלק מהעץ, ואם גדול – נסיים לעבור על עץ השירים של האמן $a_{M,1}$, נעבור לאמן הבא $a_{M,2}$ ונעשה את אותן הפעולות..

כעת נתבונן במספר הצמתים שנעבור עליהם בעץ – נסמנו $k_{M,1}$. מטענה $(*)$ נקבל כי סך הפעולות שאנו עושים הוא $O(k_{M,1})$ – והדבר נכון לגבי כל האמנים שנעבור עליהם. במקרה בו נעבור על כל האמנים שלהם יש לפחות שיר אחד עם M השמעות, נוכל לעבור לצומת הקודם לצומת "M" השמעות ב-$O(1)$ (שכן כל צומת ברשימה מחזיק מצביע לצומת הקודם לו), ולהמשיך באותו סדר פעולות.

לכן, בסופו של דבר נבצע $O(m)$ פעולות כנדרש (מטענות שראינו בכיתה):

$$O(k_{M,1}) + O(k_{M,2}) + O(k_{M,l}) + O(k_{M-p,1}) + \cdots = O(k_{M,1} + k_{M,2} + \cdots) = O(m)$$

* <u>שגיאות</u>: (אם לא היו כאלה, הפונקציה תחזיר *SUCCESS*)

- אם הייתה בעיה בהקצאת זיכרון, הפונקציה תחזיר *ALLOCATION_ERROR*.
- אם יש במבנה פחות מ-*numOfSongs* שירים (נוכל לבדוק ב-$O(1)$) – הפונקציה תחזיר *FAILURE*.
- אם *numOfSongs* או *DS* לא תקינים לפי הדרישות (נוכל לבדוק זאת ב-$O(1)$) – הפונקציה תחזיר *INVALID_INPUT*.

<u>*Quit*</u>:

תחילה ניזכר מטענה $(***)$ כי $m \geq n$, ולכן $O(m+n) = O(m)$ (מטענה שראינו בכיתה).

בעת פעולת *Quit*, אנו למעשה מוחקים את *MusicManager* ומשנים את ערך המצביע *DS* ל-*NULL* (שינוי שקורה ב-$O(1)$). אם כן, סיבוכיות הפעולה תהיה כסיבוכיות המחיקה של *MusicManager*.

בעת המחיקה, אנו קוראים להורסים של עץ האמנים ושל רשימת ההשמעות שתחת *MusicManager*.

- <u>סיבוכיות פעולת ההורס של עץ האמנים</u>:

  כל צומת בעץ מכילה אמן, ולכן מחיקת צומת תקרא להורס של האמן (ועוד $O(1)$ פעולות על מנת למחוק את הצומת כולה). בעת מחיקת אמן, אנו רוצים למחוק את עץ ההשמעות שלו וכן את מערך השירים שלו:

  - מחיקת עץ ההשמעות תיקח $O(c_i)$ פעולות, כאשר c_i מסמל את מספר השירים שיש לאמן עם i השמעות.
  - מחיקת מערך השירים הכולל תיקח $O(m_a)$ פעולות נוספות (כאשר m_a מסמל את מספר השירים הכולל של האמן), מכיוון שמדובר במערך של מצביעים בגודל m_a (את האיברים אליהם הצביעו נמחק בהמשך).

  כלומר, נבצע $O(m_a)$ פעולות למחיקת כל אמן (מטענות שראינו בכיתה):

$\sum_{i\in\{L|a\ songs\ with\ L\ streams\}} O(c_i) + O(m_a) = O(\sum_i c_i) + O(m_a) = O(m_a) + O(m_a) = O(m_a)$

בעץ הנ״ל יש n צמתים (כמספר האמנים), לכן עבור מחיקת כלל עץ האמנים נקבל:

$\sum_{a\in\{X|an\ artist\ with\ artistID=X\}} O(m_a) = O(\sum_a m_a) = O(m)$

- <u>סיבוכיות פעולת ההורס של רשימת ההשמעות</u>:

מחיקה של רשימת ההשמעות תבצע (במימוש שלנו) מחיקת צומת-צומת החל מהצומת האחרון, לצומת הלפני-אחרון וכך הלאה, עד שתמחק את כל הצמתים בה (וכן את עצמה ב-$O(1)$).

עבור כל צומת, נקרא להורס של הצומת, שלמעשה מוחק את עץ המצביעים שמכיל הצומת. פעולת המחיקה של כל עץ כזה תיקח $O(d_i)$ פעולות, כאשר d_i הוא מספר הצמתים בו (זו הסיבוכיות מכיוון שנבצע את המחיקה ע״י סיור *in-order* בו כל פעולת מחיקה היא למעשה מחיקת מצביע ב-$O(1)$).

נוכל לשים לב כי אם נסכום על כל ה-d_i-ים עבור כל הצמתים, נקבל מספר שחסום ע״י m – זאת מכיוון ש-d_i, מס׳ הצמתים בעץ המצביעים שבצומת i, למעשה מתאר את מספר האמנים שיש להם שיר עם מספר השמעות X_i כלשהו.

מכאן, ש-$\sum_i d_i \le m$.

אם נעבור על כלל הצמתים ברשימה, נקבל כי נבצע $O(d_i)$ פעולות עבור כל צומת i שמחקנו, כלומר $= O(\sum_i d_i)$ $O(m)$ פעולות כנדרש (מטענה שראינו בכיתה).

לכן, בסופו של דבר נבצע $O(m) + O(m) = O(m)$ פעולות כנדרש (מטענה שראינו בכיתה).

* <u>שגיאות</u>: לא אמורות להיות.

סיבוכיות מקום:

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

- **עץ האמנים**: ישנם n אמנים בעץ, ולכל אמן יש m_a שירים, כלומר יש m_a צמתים בכל עצי השירים שתחת האמן הנ״ל (אשר מחולקים באופן כלשהו בין הצמתים של עץ ההשמעות של אותו אמן).

  סך השירים (סכימת ה-m_a-ים של כל אמן) שתחת עץ האמנים מסתכם ל-m – מספר השירים הכולל במערכת.

  בתוספת המספר הקבוע של מקום שנשמר יחד עם כל שיר/אמן, נקבל כי סיבוכיות המקום של עץ האמנים הוא $O(c\cdot m) + O(c\cdot n) = o(m+n)$.

- **רשימת ההשמעות**: כל צומת ברשימת השמעה מכיל מזהה X (״מספר השמעות״) מספר מצביעים (אשר מסתכמים ב-$O(1)$), וכן עץ המכיל מצביעים לכל השירים במערכת שהושמעו X פעמים.

  אין שיר אליו מצביעים שני צמתים או יותר בעץ כלשהו (תחת צומת כלשהו) – מכאן שמספר הצמתים הכללי, בכל עצי המצביעים שתחת כל הצמתים ברשימה, הוא בסך הכל m.

  אם כך, סיבוכיות המקום של הרשימה הינה $O(m)$.

לכן, נקבל כי סיבוכיות המקום הכוללת היא $O(m) + O(m+n) = O(m+n)$ (מטענות שראינו בכיתה).