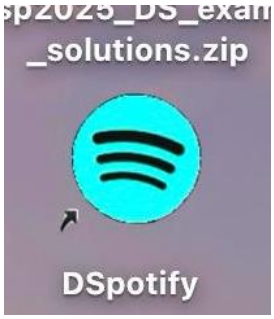
מתרגל ממונה על התרגיל: אמיר מן, amir.mann@campus.technion.ac.il

תאריך ושעת הגשה: 25/05/2025 בשעה 23:59

אופן ההגשה: בזוגות. אין להגיש ביחידים. (אלא באישור מתרגל אחראי של הקורס)

הנחיות כלליות:

- שאלות על התרגיל יש לפרסם באתר הפיאצה של הקורס תחת לשונית "hw-wet-1":
  - האתר: https://piazza.com/class/m8krba145sw2zt, נא לקרוא את השאלות של סטודנטים אחרים לפני שמפרסמים שאלה חדשה, למקרה שנשאלה כבר.
- נא לקרוא את המסמך "נהלי הקורס" באתר הקורס. בנוסף, נא לקרוא בעיון את כל ההנחיות בסוף מסמך זה.
- בפורום הפיאצה ינוהל FAQ ובמידת הצורך יועלו תיקונים כ**הודעות נעוצות** (Pinned Notes). תיקונים אלו **מחייבים**.
- התרגיל מורכב משני חלקים: יבש ורטוב.
  - לאחר קריאת כלל הדרישות, מומלץ לתכנן תחילה את מבני הנתונים על נייר. דבר זה יכול לחסוך לכם זמן רב.
  - לפני שאתם ניגשים לקודד את פתרונכם, ודאו כי יש לכם פתרון העומד בכל דרישות הסיבוכיות בתרגיל. תרגיל שאינו עומד בדרישות הסיבוכיות יחשב כפסול.
  - את הפתרון שלכם מומלץ לחלק למחלקות שונות שאפשר לממש (ולבדוק!) בהדרגתיות.
  - המלצות לפתרון התרגיל נמצאות באתר הקורס תחת: "Programming Tips Session".
    - המלצות לתכנות במסמך זה אינן מחייבות, אך מומלץ להיעזר בהן.
  - חומר התרגיל הינו כל החומר שנלמד בהרצאות ובתרגולים עד אך לא כולל עצי דרגות.
- העתקת תרגילי בית רטובים תיבדק באמצעות תוכנת בדיקות אוטומטית, המזהה דמיון בין כל העבודות הקיימות במערכת, גם כאלו משנים קודמות. לא ניתן לערער על החלטת התוכנה. התוכנה אינה מבדילה בין מקור להעתק! אנא הימנעו מהסתכלות בקוד שאינו שלכם.
- בקשות להגשה מאוחרת יש להפנות למתרגל האחראי בלבד בכתובת: jonathan.gal@campus.technion.ac.il



הקדמה:

בעקבות באג ספוטיפיי, כל פעם שמנגנים שיר הוא מתחלף ב- Boulevard of Broken Dreams של Green Day החל מדקה רנדומלית. למרות שמעטים שמחו על החידוש רוב מיליארדי המשתמשים חושבים שהאפליקציה לא שמישה יותר. סגל מבני נרתם כמובן לתיקון הבעיה ונידב אתכם לכתוב מערכת חדשה בשם DSpotify לניהול שירים ופלייליסטים שתחליף את האפליקציה האהובה.

סימונים לצורכי סיבוכיות:

נסמן ב-n את מספר השירים במערכת.

ב-m את מספר הפלייליסטים במערכת.

ב-$n_{playlistId}$ את מספר השירים שבפלייליסט שאת המזהה שלו הפונקציה מקבלת. אם אין פלייליסט כזה אפשר להתייחס לערך זה כ-1.

דרוש מבנה נתונים למימוש הפעולות הבאות:

DSpotify()

מאתחלת מבנה נתונים ריק. תחילה אין במערכת שירים או פלייליסטים.

פרמטרים: אין

ערך החזרה: אין

סיבוכיות זמן: $O(1)$ במקרה הגרוע.

---

virtual ~DSpotify()

הפעולה משחררת את המבנה (כל הזיכרון אותו הקצאתם חייב להיות משוחרר).

פרמטרים: אין

ערך החזרה: אין

סיבוכיות זמן: $O(n \cdot m)$ במקרה הגרוע.

StatusType add_playlist(int playlistId)

הפעולה מוסיפה למבנה נתונים פלייליסט ריק.

פרמטרים:

playlistId	מזהה הפלייליסט שצריך להוסיף.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם playlistId <=0.
FAILURE	אם קיים כבר פלייליסט עם מזהה playlistId.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log m)$ במקרה הגרוע.

StatusType delete_playlist(int playlistId)

הפלייליסט בעל המזהה playlistId אינו בשימוש יותר, ולכן צריך להוציאו מהמערכת.
אם קיימים שירים בפלייליסט הוא אינו יכול להימחק ונשאר במערכת.

פרמטרים:

playlistId	מזהה הפלייליסט.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם playlistId <=0.
FAILURE	אם אין פלייליסט עם מזהה playlistId או שיש בו שירים.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log m)$ במקרה הגרוע.

StatusType add_song(int $songId$, int $plays$)

שיר בעל מזהה ייחודי $songId$ מתוסף למערכת, הוא אינו באף פלייליסט, ושמעו אותו plays פעמים.

פרמטרים:

$songId$	מזהה השיר שצריך להוסיף.
$plays$	מספר ההשמעות של השיר שצריך להוסיף.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם $songId <=0$ או אם $plays <0$.
FAILURE	אם קיים כבר שיר במזהה $songId$ במערכת.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log n)$ במקרה הגרוע.

StatusType add_to_playlist(int $playlistId$, int $songId$)

השיר בעל המזהה $songId$ מצטרף לפלייליסט בעל המזהה $playlistId$, במידה ולפני כן לא היה בפלייליסט זה.

פרמטרים:

$songId$	מזהה השיר שמתוסף לפלייליסט.
$playlistId$	מזהה הפלייליסט אליו השיר מתוסף.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם $songId <=0$ או אם $playlistId <= 0$.
FAILURE	אם אין שיר עם מזהה $songId$ השיר הזה כבר נמצא בפלייליסט במזהה $playlistId$, או שאין פלייליסט במזהה $playlistId$.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log n + \log m)$ במקרה הגרוע.

StatusType delete_song(int $songId$)

השיר בעל המזהה songId אינו בשימוש יותר, ולכן צריך להוציאו מהמערכת.
אם השיר נמצא בפלייליסט כלשהוא הפעולה נכשלת והמערכת לא משתנה.

פרמטרים:

songId	מזהה השיר.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם songId <=0.
FAILURE	אם אין שיר עם מזהה songId או שהוא נמצא בפלייליסט כלשהוא.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log n)$ במקרה הגרוע.

StatusType remove_from_playlist(int $playlistId$, int $songId$)

השיר בעל המזהה $songId$ מוצא מהפלייליסט בעל המזהה $playlistId$.

פרמטרים:

$songId$	מזהה השיר שמוציאים מהפלייליסט.
$playlistId$	מזהה הפלייליסט ממנו יש להוציא שיר.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם $songId<=0$ או אם $playlistId <= 0$.
FAILURE	אם אין פלייליסט במזהה $playlistId$ או אם השיר במזהה $songId$ אינו בפלייליסט שבמזהה $playlistId$.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log m + \log n_{playlistId})$ במקרה הגרוע.

---

output_t $< int >$ get_plays(int $songId$)

מחזיר את מספר ההשמעות של השיר במזהה $songId$.

פרמטרים:

$songId$	מזהה השיר שאת מהירותו יש להחזיר.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם $songId<=0$.
FAILURE	אם אין שיר במזהה $songId$.
SUCCESS	במקרה של הצלחה, במקרה זה תוחזר גם מספר ההשמעול של השיר.

סיבוכיות זמן: $O(\log n)$ במקרה הגרוע.

---

output_t $< int >$ get_num_songs(int $playlistId$)

מחזיר את מספר השירים שבפליילסט בעל המזהה $playlistId$.

פרמטרים:

$playlistId$	מזהה הפלייליסט שאת מספר השירים שלו יש להחזיר.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם $playlistId<=0$.
FAILURE	אם אין פלייליסט במזהה $playlistId$.
SUCCESS	במקרה של הצלחה, במקרה זה יוחזר גם מספר השירים בפלייליסט.

סיבוכיות זמן: $O(\log m)$ במקרה הגרוע.

---

output_t < int > get_by_plays(int $playlistId$, int $plays$)

מבין השירים שבפליילסט בעל המזהה $playlistId$. מחזיר את מזהה השיר שהושמע plays פעמים. אם אין כזה יוחזר השיר שהושמע המספר הקרוב ביותר ל-$plays$ פעמים מלמעלה. אם ישנם מספר שירים בעלי אותו מספר $plays$ שעומדים בהגדרה זו יוחזר האחד עם המזהה הנמוך ביותר.

<u>פרמטרים</u>:

$playlistId$	מזהה הפלייליסט שבו יש לחפש.
$plays$	מספר ההשמעות שיש לחפש בפליילסט.

<u>ערך החזרה</u>:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם $plays$ <0 או אם $playlistId$ <=0.
FAILURE	אם אין פלייליסט במזהה $playlistId$ או שאין אף שיר בעל לכל הפחות plays השמעות בפלייליסט.
SUCCESS	במקרה של הצלחה, במקרה זה יוחזר גם מזהה השיר בעל מספר השמעות מינימלי שגדול שווה מ-plays. במקרה של מספר שירים העונים על הגדרה זו יוחזר האחד עם המזהה המינימלי.

<u>סיבוכיות זמן</u>: $O(\log m + \log n_{playlistId})$ במקרה הגרוע.

StatusType $unite\_playlists$(int $playlistId1$, int $playlistId2$)

מאחד בין שני הפלייליסטים במזהים $playlistId1$ ו-$playlistId2$. הפלייליסט במזהה $playlistId2$ נמחק לאחר הפעולה ואינו קיים יותר. הפלייליסט המאוחד נשמר במזהה $playlistId1$.

<u>פרמטרים</u>:

$playlistId1$	מזהה הפלייליסט אליו מוסיפים את הפלייליסט השני.
$playlistId2$	מזהה הפליילסט שמוסיפים לפלייליסט אחר, ולאחר מכן מוחקים.

<u>ערך החזרה</u>:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם $playlistId1$ <=0 או אם $playlistId2$ <=0 או שאלה אותם מזהים כלומר מתקיים $playlistId1==playlistId2$.
FAILURE	אם אין פלייליסט במזהה $playlistId1$ או שאין פלייליסט במזהה $playlistId2$.
SUCCESS	במקרה של הצלחה.

<u>סיבוכיות זמן</u>: $O(\log m + n)$ במקרה הגרוע.

דוגמה הרצה:

add_playlist(1): SUCCESS
add_playlist(2): SUCCESS
add_song(10, 5): SUCCESS
add_song(20, 8): SUCCESS
add_song(30, 2): SUCCESS
add_to_playlist(10, 1): SUCCESS
add_to_playlist(20, 1): SUCCESS
add_to_playlist(30, 2): SUCCESS
remove_from_playlist(1, 10): SUCCESS
get_plays(10): SUCCESS, Value: 5
delete_song(10): SUCCESS
add_song(10, 4): SUCCESS
add_to_playlist(10, 2): SUCCESS
add_plays(10, 6): SUCCESS
get_plays(10): SUCCESS, Value: 10
get_plays(20): SUCCESS, Value: 8
get_num_songs(1): SUCCESS, Value: 1
get_num_songs(2): SUCCESS, Value: 2
get_by_plays(2, 5): SUCCESS, Value: 10
get_by_plays(2, 2): SUCCESS, Value: 30
get_by_plays(1, 5): SUCCESS, Value: 20

סיבוכיות מקום:

סיבוכיות המקום הדרושה עבור מבנה הנתונים היא $O(n + m + \sum_{i \text{ is a playlistId}} n_i)$ במקרה הגרוע. כלומר בכל רגע בזמן הריצה, צריכת המקום של מבנה הנתונים תהיה לינארית בסכום מספרי השירים, הפלייליסטים במערכת ומספר השירים בכל הפלייליסטים יחד כולל כפילויות.

סיבוכיות המקום הנדרשת עבור כל פעולה (כלומר, זיכרון "העזר" שכל פעולה משתמשת בו) אינה מצוינת לכל פעולה לחוד, אך אסור לעבור את סיבוכיות המקום הדרושה שמוגדרת לכל המבנה.

ערכי החזרה של הפונקציות:

כל אחת מהפונקציות מחזירה ערך מטיפוס StatusType שייקבע לפי הכלל הבא:

- תחילה, יוחזר INVALID_INPUT אם הקלט אינו תקין.
- אם לא הוחזר INVALID_INPUT:
  - בכל שלב בפונקציה, אם קרתה שגיאת הקצאה/שחרור יש להחזיר ALLOCATION_ERROR. מצב זה אינו צפוי אלא באחד משני מקרים (לרוב): באמת השתמשתם בקלט גדול מאוד ולכן המבנה ניצל את כל הזיכרון במערכת, או שיש זליגת זיכרון בקוד.
  - אם קרתה שגיאה אחרת, כפי שמצוין בכל פונקציה, יש להחזיר מיד FAILURE <u>מבלי</u> לשנות את מבנה הנתונים.
- אחרת, יוחזר SUCCESS.

חלק מהפונקציות צריכות להחזיר בנוסף עוד פרמטר (int או bool), לכן הן מחזירות אובייקט מטיפוס output_t<T>.
אובייקט זה מכיל שני שדות: הסטטוס (__status) ושדה נוסף (__ans) מסוג T.
במקרה של הצלחה (SUCCESS), השדה הנוסף יכיל את ערך החזרה, והסטטוס יכיל את SUCCESS. בכל מקרה אחר, הסטטוס יכיל את סוג השגיאה והשדה הנוסף לא מעניין.

שני הטיפוסים (output_t<T>,StatusType) ממומשים כבר בקובץ "wet1util.h" שניתן לכם כחלק מהתרגיל.

קיים קונסטרקטור של output_t<T> מ-T ומ-StatusType כך שניתן פשוט לכתוב בפונקציות הרלוונטיות:

```
return 7;
return StatusType::FAILURE;
```

<u>הנחיות ודגשים כלליים:</u>

<u>חלק יבש:</u>

- החלק היבש הווה חלק מהציון על התרגיל כפי שמצוין בנהלי הקורס.
- לפני מימוש הפעולות בקוד יש לתכנן היטב את מבני הנתונים והאלגוריתמים ולוודא כי באפשרותכם לממש את הפעולות בדרישות הזמן והזיכרון שלעיל.
- החלק היבש חייב להיות מוקלד.
- הגשת החלק הרטוב מהווה תנאי הכרחי לקבלת ציון על החלק היבש, כלומר, הגשה בה יתקבל אך ורק חלק יבש תגרור ציון 0 על התרגיל כולו.
- יש להכין מסמך הכולל תיאור של מבני הנתונים והאלגוריתמים בהם השתמשתם בצירוף הוכחת סיבוכיות הזמן והמקום שלהם. חלק זה עומד בפני עצמו וצריך להיות מובן לקורא גם לפני העיון בקוד. אין צורך לתאר את הקוד ברמת המשתנים, הפונקציות והמחלקות, אלא ברמה העקרונית. חלק יבש זה <u>לא</u> תיעוד קוד.
- ראשית הציגו את מבני הנתונים בהם השתמשתם. רצוי ומומלץ להיעזר בציור.
- לאחר מכן הסבירו כיצד מימשתם כל אחת מהפעולות הנדרשות. הוכיחו את דרישות סיבוכיות הזמן של כל פעולה תוך כדי התייחסות לשינויים שהפעולות גורמות במבני הנתונים.
- הוכיחו שמבנה הנתונים וכל הפעולות עומדים בדרישת סיבוכיות המקום.
- החסמים הנתונים בתרגיל הם לא בהכרח הדוקים ולכן יכול להיות שקיים פתרון בסיבוכיות טובה יותר. מספיק להוכיח את החסמים הדרושים בתרגיל.
- רמת פירוט: יש להסביר את כל הפרטים שאינם טריוויאליים ושחשובים לצורך מימוש הפעולות ועמידה בדרישות הסיבוכיות. אין לדון בפרטים טריוויאליים (הפעילו את שיקול דעתכם בקשר לזה, ושאלו את האחראי על התרגיל אם אינכם בטוחים). אין לצטט קטעים מהקוד כתחליף להסבר. אין צורך לפרט אלגוריתמים שנלמדו בכתה. כמו כן, אין צורך להוכיח תוצאות ידועות שנלמדו בכתה, אלא מספיק לציין בבירור לאיזו תוצאה אתם מתכוונים. אין (וגם אין צורך) להשתמש בתוצאות של עצי דרגות והלאה.
- **על חלק זה לא לחרוג מ-8 עמודים.**
- והכי חשוב **keep it simple!**

<u>חלק רטוב:</u>

- <u>מומלץ לממש תחילה את מבני הנתונים בצורה הכללית ביותר ורק אז לממש את הפונקציות הנדרשות בתרגיל.</u>
- אנו ממליצים בחום על מימוש **Object Oriented**, ב**++C**, מימוש כזה יאפשר לכם להגיע לפתרון פשוט וקצר יותר לפונקציות אותן עליכם לממש ויאפשר לכם להכליל בקלות את מבני הנתונים שלכם (זכרו שיש תרגיל רטוב נוסף בהמשך הסמסטר).
- פקודת הקימפול שמורצת בgradescope הינה: g++ -std=c++14 -DNDEBUG -Wall -o main.out *.cpp
- חתימות הפונקציות שעליכם לממש ומספר הגדרות נמצאים בקובץ plains25b1.h.
- אין לשנות את הקבצים main25b1.cpp ו-wet1util.h אשר סופקו כחלק מהתרגיל, ואין להגיש אותם. ישנה בדיקה אוטומטית שאין בקוד שימוש בSTL, ובדיקה זו נופלת אם מגישים גם את main25b1.cpp.
- את שאר הקבצים ניתן לשנות, ותוכלו להוסיף קבצים נוספים כרצונכם, ולהגיש אותם.
- העיקר הוא שהקוד שאתם מגישים יתקמפל עם הפקודה לעיל, כאשר מוסיפים לו את שני הקבצים main25b1.cpp ו-wet1util.h.
- עליכם לממש בעצמכם את כל מבני הנתונים (למשל **אין להשתמש במבנים של STL** ואין להוריד מבני נתונים מהאינטרנט). **כחלק מתהליך הבדיקה אנו נבצע בדיקה ידנית של הקוד ונוודא שאכן מימשתם את מבני הנתונים שבהם השתמשתם.**
  - בפרט, אסור להשתמש ב-std::iterator, std::vector, std::pair, או כל אלגוריתם של STL, רשימה מלאה של הספריות להן אסור לעשות include נמצאת בקובץ dont_include.txt.
  - **ניתן** להשתמש במצביעים חכמים (Smart pointers כמו shared_ptr), בספריית math או בספריית exception.
  - חשוב לוודא שאתם מקצים/משחררים זיכרון בצורה נכונה (מומלץ לוודא עם valgrind). לא חייבים לעבוד עם מצביעים חכמים, אך אם אתם מחליטים כן לעשות זאת, לוודא שאתם משתמשים בהם נכון. (תזכרו שהם לא פתרון קסם, למשל, כאשר יוצרים מעגל בהצבעות). שימו לב שהעתקת מצביעים חכמים היא איטית מאד ולכן יכולה לגרום ל-timeout. עדיף להעביר shared_ptr כשצריכים by reference.

- שגיאות של ALLOCATION_ERROR בד"כ מעידות על זליגה בזיכרון.
- על הקוד להתקמפל ולעבור את כל הבדיקות שמפורסמות לכם ב-gradescope. הטסטים שמורצים באתר מייצגים את הבדיקת אותן נריץ בנתינת הציון, כאשר פרסמנו 5 מתוך 50.
- אותם טסטים שבgradescope גם מפורסמים כקבצי קלט ופלט, יחד עם סקריפט בשם run_tests.py שנכתב בשביל python 3.6 ומעלה, המאפשר לבדוק את הקוד שלכם. מומלץ לבדוק את התרגיל לוקאלית לפני שמגישים.
- במידה ויש timeout על אחד מהטסטים זה יחשב ככשלון בטסט. עליכם לכתוב קוד יעיל במידת הסביר. אין לדאוג מכך יותר מידי, הרוב המוחלט של הפתרונות שעומד בסיבוכיות עומד גם בזמני הריצה.
- **שימו לב:** התוכנית שלכם תיבדק על קלטים רבים ושונים מקבצי הדוגמא הנ"ל. יחד עם זאת הטסטים האלו מייצגים מבחינת אורך ואופן היצירה שלהם את השאר.

אופן ההגשה:

הגשת התרגיל הנה דרך אתר ה-gradescope של הקורס.

חלק הרטוב:

יש להגיש רק את קבצי הקוד שלכם (לרוב קבצי cpp. ,h. בלבד אפשר גם להגיש אותם כקובץ zip)

חלק היבש:

יש להגיש קובץ PDF אשר מכיל את הפתרון היבש. החלק היבש חייב להיות מוקלד.

- **שימו לב כי אתם מגישים את כל שני החלקים הנ"ל, במטלות השונות.**
- לאחר שהגשתם, יש באפשרותכם לשנות את התוכנית ולהגיש שוב. ההגשה האחרונה היא הנחשבת.
- הערכת הציון שמופיעה ב-gradescope אינה ציונכם הסופי על המטלה. הציון הסופי יתפרסם רק לאחר ההגשות המאוחרות של משרתי המילואים.
- במידה ואתם חושבים שישנה תקלה מהותית במערכת הבדיקה ב-gradescope נא להעלות זאת בפורום הפיאצה ונתפל בה בהקדם.

דחיות ואיחורים בהגשה:

- דחיות בתרגיל הבית תינתנה אך ורק לפי תקנון הקורס.
- 5 נקודות יורדו על כל יום איחור בהגשה ללא אישור מראש. באפשרותכם להגיש תרגיל באיחור של עד 5 ימים ללא אישור. תרגיל שיוגש באיחור של יותר מ-5 ימים ללא אישור מראש יקבל 0.
- במקרה של איחור בהגשת התרגיל יש עדיין להגיש את התרגיל אלקטרונית דרך אתר הקורס.
- בקשות להגשה מאוחרת יש להפנות למתרגל האחראי בלבד בכתובת jonathan.gal@campus.technion.ac.il. לאחר קבלת אישור במייל על הבקשה, מספר הימים שאושרו לכם נשמר אצלנו. לכן, אין צורך לצרף להגשת התרגיל אישורים נוספים או את שער ההגשה באיחור.

בהצלחה!