האוניברסיטה העברית בירושלים

בית הספר להנדסה ולמדעי המחשב ע"ש רחל וסלים בנין

סדנת תכנות בשפת C ו־++C (67320) - קורס קיץ

תרגיל 5 ־ שפת ++C

תאריך הגשה: 07.09.22 בשעה 22:00

נושאי התרגיל: שימוש במבני נתונים בספריית STL, אלגוריתמים של STL ומצביעים חכמים.
המסמך אמנם ארוך אך ברובו מכיל דוגמות, חישובים מפורטים והגדרות מתמטיות.

1 רקע

בעבודתכם החדשה ראו שאתם מתכנתי CPP נהדרים ולכן ביקשו מכם לממש פרויקט בעצמכם. מטרת הפרויקט היא ליצור ספריה שתשמש את חברות הזרמת המדיה (Streaming) הגדולות. הספרייה היא כלי ניהול של מערכות המלצה ומשתמשים, על מנת לאפשר להן להמליץ ללקוחותיהן על הסרטים המתאימים ביותר. עליכם לממש את הספרייה כפי שמוגדר להלן.

2 בניית המשתמשים ומערכת ההמלצות

2.1 מבנה הספרייה

הספרייה תורכב מחמש מחלקות:

1. Movie ־ מחלקה זו אחראית על ייצוג סרט במערכת
2. RSUser ־ מחלקה המייצגת משתמש של המערכת
3. RecommenderSystem ־ מחלקה הנותנת המלצות צפיה עבור משתמש מסוים
4. RecommenderSystemLoader ־ מחלקה האחראית לייצור מערכת ההמלצה
5. RSUsersLoader ־ מחלקה האחראית לייצור המשתמשים

שלד עבור תרגיל זה נמצא במודל. עליכם להשתמש בו ולהוסיף את ה־API הדרוש, אך **אין לשנות את קובץ השלד**.

שימו ❤️ : תעברו על הקוד הנתון לכם לפני תחילת כתיבת הקוד.

2.2 מחלקת Movie

נוסף על ה־API המפורט מטה שעליכם לממש, עליכם להגדיר בקובץ ה־h של המחלקה typedef הנקרא sp_movie ה־typedef יהיה של מצביע מסוג Movie. בחרו מצביע כך שאין צורך בניהול זיכרון ומשתמשים שונים יכולים להצביע על אותו המקום בזיכרון של סרט כלשהו.

שימו לב כי בהמשך המסמך ובטסטים נשתמש בהגדרות האלו, ולכן חשוב שתממשו אותם (חוסר מימוש שלהם עלול לגרום לכישלון הטסטים).

Constructor	הבנאי מקבל מחרוזת המייצגת את שם הסרט ומספר המייצג את שנת ההוצאה שלו
operator <<	אופרטור שמעביר ל־ostream את פרטי הסרט בפורמט הבא: <movie_name> (<movie_year>)\n
operator <	אופרטור השוואה > ביחס לשנת ההוצאה של הסרט. אם השנים זהות האופרטור מחזיר true אם שם הסרט השמאלי קטן יותר לפי סדר לקסיקוגרפי כדי להבהיר את הגדרת האופרטור > עבור המחלקה movie, נתבונן בדוגמה הבאה. עבור הסרטים Twilight-2008, Titanic-1997, Wanted-2008, מתקיימים היחסים הבאים (על פי ההגדרה בתרגיל): Twilight-2008 > Titanic-1997 מכיוון ששנת היציאה של הטיטאניק קטנה משנת היציאה של דמדומים. כמו כן, Wanted-2008 > Twilight-2008 מכיוון ששניהם יצאו באותה שנה אך "Twilight" קטן יותר (בסדר לקסיקוגרפי) מ-"Wanted".
get_name	הפונקציה תחזיר את שם הסרט
get_year	הפונקציה תחזיר את שנת ההוצאה של הסרט

טבלה 1: Movie API

ניתן להניח את תקינות הקלט. שימו לב כי שני סרטים a, b נחשבים זהים אם כל אחד מהם לא קטן מהשני, כלומר $!(a < b) \wedge !(b < a)$

ניתן להניח שלא יהיה עוד מקף ('-') בשמות של הסרטים.

שימו לב כי חלק מה־API אינו כולל את החתימות המפורשות. עליכם להשלים את החתימה בצורה מתאימה, בהתאם למה שראיתם בהרצאות ובתרגולים.

בקובץ השלד שניתן לכם עבור המחלקה, יש מספר הגדרות חשובות. אסור למחוק אותן ועליכם להשתמש בהן במהלך כתיבת התרגיל

```
typedef std::size_t (*hash_func)(const sp_movie& movie);
typedef bool (*equal_func)(const sp_movie& m1,const sp_movie& m2);
std::size_t sp_movie_hash(const sp_movie& movie);
bool sp_movie_equal(const sp_movie& m1,const sp_movie& m2);
```

לשתי הפונקציות האחרונות יש מימוש בקובץ Movie.cpp. אין לשנות או להדיד את המימוש - שינוי או הדה עלול לגרום לכישלון הטסטים.

2.3 מחלקת RSUser

מחלקה זו מייצגת משתמש אחד.

[1] לידיעתכם, יש מספר דרכים לממש מצביע לפונקציה - std::function, lambda, אובייקטים ועוד. מי שרוצה להרחיב על std::function מוזמן לקרוא למשל כאן https://en.cppreference.com/w/cpp/utility/functional/function

בשלד של המחלקה הדאת מוגדר לכם ה־typedef הבא, ועליכם להשתמש בו במהלך מימוש ה־API:

```
typedef std::unordered_map<sp_movie, double, hash_func, equal_func> rank_map;
```

Constructor	הבנאי מקבל מחרוזת המייצגת שם משתמש, את הדירוגים שלו לסרטים שהוא ראה ומצביע למערכת המלצה. החתימה המדויקת של הבנאי ניתנת לבחירתכם
get_name()	הפונקציה מחזירה את שם המשתמש
void add_movie_to_rs(const std::string &name, int year, const std::vector<double>& features, double rate)	הפונקציה מקבלת מחרוזת המייצגת את שם הסרט, מספר צג את שנת ההוצאה, את ערכי התכונות השונות שלו ואת דירוג המשתמש לסרט זה. היא מוסיפה אותו למאגר של מערכת ההמלצה (שימו לב כי שינוי זה משפיע על כל המשתמשים של מערכת ההמלצה) ושומרת את ערכי התכונות שלו. המחלקות RecommenderSystemLoader, RSUsersLoader מאתחלות את המשתמשים ואת מערכת ההמלצה מתוך הקבצים המתאימים. המטרה של הפונקציה add_movie של מחלקת RSUser היא לתת אפשרות למשתמש להרחיב את המאגר לאחר יצירתו, באמצעות הוספה של סרטים נוספים שבהם הוא צפה. כדי לעשות זאת הוא נדרש לספק את הפרטים של הסרט, ואת הדירוג שלו, על מנת לשפר את החיזויים עבורו. אפשר להניח שהוספת סרטים היא עקבית עם המאגר שלה, כלומר אם במהלך יצירת המאגר היו 5 פיצ'רים לסרטים, כאשר נוסיף סרט חדש יהיה גם לו 5 פיצ'רים. בנוסף, ניתן להניח שלא ניתן ל add_movie-סרט שכבר קיים במערכת (כלומר בעל אותו שם ושנה)
sp_movie get_recommendation_by_content() const	מתודה זו תחזיר מצביע לסרט המומלץ לפי אלגוריתם המלצה לפי תוכן.
double get_prediction_score_for_movie(const std::string& name, int year, int k)	הפונקציה מקבלת מחרוזת של שם הסרט עבורו רוצים לחזות את הדירוג, את שנת ההוצאה שלו, ו־*k* מספר שלם וחיובי (הפרמטר באלגוריתם הסינון השיתופי) המייצג את מספר הסרטים הדומים ביותר (ומדורגים על ידי המשתמש) לסרט, עליהם נתבסס בחיזוי. הפונקציה מחזירה את חיזוי הדירוג של המשתמש עבור הסרט לפי שיטת הסינון השיתופי. ניתן להניח אותן הנחות כמו ב־RecommenderSystem
sp_movie get_recommendation_by_cf(int k) const	מתודה זו תקבל מספר שלם וחיובי (הפרמטר באלגוריתם הסינון השיתופי) המייצג את מספר הסרטים הדומים ביותר לכל סרט (ומדורגים על ידי המשתמש) עליהם נתבסס בחיזוי. המתודה מחזירה מצביע לסרט עליו נמליץ למשתמש לפי שיטת סינון שיתופי כפי שהוסבר לעיל. ניתן להניח כי הפרמטר *k* שלם וחיובי, וקטן ממספר הסרטים שדורגו על ידי המשתמש.

operator <<	אופרטור שמעביר לostream את שם המשתמש בפורמט "name: <NAME>\n", ובשורה הבאה את כל הסרטים במאגר המתאים לו בצורה ממוינת (לפי האופרטור <), על פי פורמט ההדפסה של הסרט. אחרי הדפסת כל הסרטים יש להוסיף endl. בהדפסת RSUser מדפיסים גם את הסרטים שהוא לא ראה במערכת ההמלצות (כלומר מדפיסים את כל מאגר הסרטים שמקושר אליו)
get_ranks()	הפונקציה מחזירה את הדירוגים של המשתמש עצמו(כלומר מחזירה את האובייקט מסוג rank_map)

טבלה 2: RSUser API

- אין צורך לשמור RS שכבר נוצרו מRSLoader בשביל לחסוך קריאות IO. **אם קיבלתם את אותו הקובץ בשתי קריאות שונות, יש ליצור שני אובייקטים חדשים שאינם קשורים האחד לשני.**

2.4 מחלקת RecommenderSystem

מחלקה זו אחראית על הלוגיקה של מערכת ההמלצות של המשתמש. מערכת ההמלצה היא מערכת כבדה **ולכן אסור להעתיק אותה**. (מה המשפט הזה **רומז** לנו?)

שימו לב כי קובץ ה־h עם חתימות הפונקציות נמצא במודל.

הסבר על שיטות ההמלצה מופיע מיד לאחר ה־API (2.4.1, 2.4.2)
נגדיר את ה־API של המחלקה:

Constructor	בנאי שאינו מקבל פרמטרים
sp_movie get_recommendation_by_content (const RSUser& user_rankings)	מתודה זו תחזיר מצביע חכם לסרט המומלץ לפי אלגוריתם המלצה לפי תוכן. ניתן להניח שהמתודה תיקרא רק עבור משתמשים שדירגו יותר מסרט אחד עם דירוגים שונים.
double predict_movie_score(const RSUser& user_rankings, const sp_movie &movie, int k)	מתודה זו תקבל אובייקט של משתמש, מצביע לסרט עבורו רוצים לחזות את הדירוג, *k* מספר שלם וחיובי (הפרמטר באלגוריתם הסינון השיתופי) המייצג את מספר הסרטים הדומים ביותר (ומדורגים על ידי המשתמש) לסרט, עליהם נתבסס בחיזוי. המתודה מחזירה את חיזוי הדירוג של המשתמש עבור הסרט לפי שיטת הסינון השיתופי. ניתן להניח כי הפרמטר *k* קטן ממספר הסרטים שדורגו על ידי המשתמש. ניתן להניח כי הסרט קיים במערכת, ושהמשתמש לא דירג אותו.
sp_movie recommend_by_cf(const RSUser& user_rankings, int k)	מתודה זו תקבל אובייקט של משתמש ומספר שלם וחיובי (הפרמטר באלגוריתם הסינון השיתופי) המייצג את מספר הסרטים הדומים ביותר לכל סרט (ומדורגים על ידי המשתמש) עליהם נתבסס בחיזוי. המתודה מחזירה את הסרט עליו נמליץ למשתמש לפי שיטת סינון שיתופי כפי שהוסבר לעיל. ניתן להניח כי הפרמטר *k* שלם וחיובי, וקטן ממספר הסרטים שדורגו על ידי המשתמש.
sp_movie add_movie(const std::string& name,int year, const std::vector<double>& features)	הפונקציה מקבלת מחרוזת של שם הסרט, שנת ההוצאה שלו ואת ערכי התכונות השונות שלו ומוסיפה אותו למאגר. הפונקציה תחזיר מצביע לסרט. ניתן להניח שהסרט לא נמצא במערכת וגודל התכונות שמתקבל זהה לגודל שקיים במערכת. דרישת היעילות של פונקציה זו היא $O(log(n))$ כאשר n הוא מספר הסרטים במערכת.
sp_movie get_movie(const std::string& name, int year) const	הפונקציה תחזיר מצביע חכם לסרט עם השם והשנה שהיא מקבלת. המצביע צריך להיות זהה למצביע שנמצא במערכת ההמלצה. אם הסרט לא קיים במערכת יש להחזיר sp_movie ל־nullptr. דרישת היעילות של פונקציה זו היא $O(log(n))$ כאשר n הוא מספר הסרטים במערכת.
operator <<	אופרטור שמעביר ל־ostream את כל הסרטים במאגר בצורה ממוינת (לפי האופרטור <), על פי פורמט ההדפסה של הסרט. דרישת היעילות של פונקציה זו היא $O(n)$ כאשר n הוא מספר הסרטים במערכת

טבלה 3: RecommenderSystem API

עליכם לממש את המחלקה בצורה יעילה, ובפרט עליכם לעמוד בדרישות היעילות שניתנות בחלק מהפונקציות.

עליכם לחשוב על מבנה נתונים מתאים מתוך *STL* שיאפשר מימוש העומד ביעילות הנדרשת.

כדי שמבנה הנתונים שבחרתם יעמוד בדרישות, **ניתן ואף מומלץ לקרוא ולהיעזר בקוד ובמבנה הנתונים שסופק לכם ב־RSUser לשם השראה**, ולהשתמש בו עם שינויים מתאימים.

2.4.1 <u>המלצה לפי תוכן</u>

רעיון כללי - המלצה על סרטים שדומים למה שהמשתמש דירג גבוה. נרצה להמליץ לו על סרט שאנו מאמינים שיאהב.

שלב 1: נחסיר את ממוצע הדירוגים של משתמש x מהדירוגים שלו, כדי לנרמל את הדירוגים.

שלב 2: ניצור וקטור העדפה של תכונות למשתמש x המורכב מהדירוגים שלו לסרטים, ביחד עם תכונותיהם של אותם סרטים.

שימו לב כי וקטור התוצאה משלב 2 בקאורדינטה ה-i מייצג את המשקל שמשתמש x נותן לתכונה ה-i , כלומר כמה הוא "אוהב" את התכונה ה-i.

שלב 3: נחשב את הדמיון על ידי חישוב הזווית בין וקטור ההעדפה של משתמש x לבין כל אחד מוקטורי התכונות של הסרטים אותם משתמש x לא דירג - ונמליץ על הסרט עם הדמיון המקסימלי בתכונות.

דוגמא:
נרצה להמליץ לSofia על סרט לפי שיטת המלצה לפי תוכן.

<u>שלב 1</u>

וקטור הדירוגים של סופיה הוא:

	Titanic (1997)	Twilight (2008)	ForestGump (1994)	Batman (2022)	StarWars (1977)
Sofia	4	NA	8	NA	NA

- ממוצע הוקטור הוא $\frac{4+8}{2} = 6$.

שימו לב כי לא התייחסנו לערכים הריקים בחישוב הממוצע.

נקבל כי וקטור הדירוגים המנורמל של סופיה הוא:

	Titanic (1997)	Twilight (2008)	ForestGump (1994)	Batman (2022)	StarWars (1977)
Sofia	-2	NA	2	NA	NA

שלב 2

ניצור את ווקטור העדפה של Sofia

1. הדירוג המנורמל של Sofia ל־Titanic הוא -2, ווקטור התכונות של טיטאניק יהיה:

Titanic (1997)	7	2	9	1

2. הדירוג המנורמל של Sofia לForestGump הוא 2, ו־וקטור התכונות של ForestGump הוא:

ForestGump (1994)	1	7	7	6

3. נקבל כי בסה"כ וקטור ההעדפות של סופיה הוא:

$$-2\cdot\begin{pmatrix}7\\2\\9\\1\end{pmatrix}+2\cdot\begin{pmatrix}1\\7\\7\\6\end{pmatrix}=\begin{pmatrix}-12\\10\\-4\\10\end{pmatrix}$$

אינטואיציה: סופיה אוהבת מאוד סרטים מפתיעים ומצחיקים (באותה מידה),לא אוהבת סרטים דרמטיים ומאוד לא אוהבת סרטים מפחידים.

שלב 3

חישוב הדמיון בין וקטור ההעדפות של סופיה לוקטורי התכונות של הסרטים ש־Sofia לא דירגה ־ Twilight, Batman, StarWars.

1. וקטור התכונות של Twilight :

Twilight (2008)	3	4	6	5

כלומר, הדמיון בין התכונות שלו לבין ההעדפות של Sofia הוא:

$$\frac{\begin{pmatrix}-12\\10\\-4\\10\end{pmatrix}\cdot\begin{pmatrix}3\\4\\6\\5\end{pmatrix}}{\left\|\begin{pmatrix}-12\\10\\-4\\10\end{pmatrix}\right\|\cdot\left\|\begin{pmatrix}3\\4\\6\\5\end{pmatrix}\right\|}=\frac{30}{\sqrt{360}\cdot\sqrt{86}}=0.17$$

2. וקטור התכונות של Batman :

Batman (2022)	2	6	4	8

כלומר, הדמיון בין התכונות שלו לבין ההעדפות של So a הוא:

$$\frac{\begin{pmatrix}-12\\10\\-4\\10\end{pmatrix}\cdot\begin{pmatrix}2\\6\\4\\8\end{pmatrix}}{\left\|\begin{pmatrix}-12\\10\\-4\\10\end{pmatrix}\right\|\cdot\left\|\begin{pmatrix}2\\6\\4\\8\end{pmatrix}\right\|}=\frac{100}{\sqrt{360}\cdot\sqrt{120}}=0.48$$

3. וקטור התכונות של StarWars :

StarWars (1977)	3	3	4	9

כלומר, הדמיון בין התכונות שלו לבין ההעדפות של So a הוא:

$$\frac{\begin{pmatrix}-12\\10\\-4\\10\end{pmatrix}\cdot\begin{pmatrix}3\\3\\4\\9\end{pmatrix}}{\left\|\begin{pmatrix}-12\\10\\-4\\10\end{pmatrix}\right\|\cdot\left\|\begin{pmatrix}3\\3\\4\\9\end{pmatrix}\right\|}=\frac{68}{\sqrt{360}\cdot\sqrt{115}}=0.33$$

מסקנה: נמליץ לSofia על הסרט Batman מכיוון שוקטור התכונות שלו הכי דומות להעדפות של Sofia

2.4.2 <u>המלצה לפי סינון</u>

רעיון כללי:
נרצה לתת המלצה למשתמש על סרט שהוא לא ראה, בהסתמך על הסרטים שהוא ראה שהם הדומים ביותר לאותו הסרט שלא ראה.
כלומר, עבור סרט m, נמצא סט $N = \{N_1, .., N_k\}$ המכיל k סרטים שהכי דומים לסרט m <u>וגם</u> שהמשתמש דירג.
לפי זה, <u>נחזה (predict) את הדירוג של משתמש x</u> עבור הסרט m.
נעשה זאת באמצעות פונקציה הלוקחת את הדירוגים של המשתמש ה-x עבור אותם $N_1, .., N_k$ הסרטים בסט, ומחשבת את הדימיון (זוויות) בין כל N_j , $j \in [1, k]$ לסרט m.

על מנת לחזות את הדירוג של משתמש x עבור סרט m, נפעל בצורה הבאה:

<u>שלב 1:</u>

נמצא את הסט $N = \{N_1, .., N_k\}$ של k הסרטים שהכי דומים לסרט m וגם שמשתמש x דירג.

שלב 2:

נחזה את הדירוג של משתמש x לסרט הm באופן הבא:

$$r_{x,m} = \frac{\sum_{j \in N} s_{m,j} \cdot r_{x,j}}{\sum_{j \in N} s_{m,j}}$$

כאשר $s_{m,j}$ הוא הדמיון בין הסרט הm לסרט הj , ו$r_{x,m}$ הוא הדירוג של המשתמש x עבור הסרט m.

כלומר, על מנת להמליץ למשתמש x על סרט, נוכל לחזות את הדירוג שלו עבור כל סרט אותו לא דירג, ולהמליץ לו על הסרט בעל הדירוג הגבוה ביותר שחזינו.

- אפשר להניח שלא ניתן לכם מקרה בו תצטרכו לחלק ב0 כאשר מנסים לחזות ציון של סרט עבור משתמש כלשהו

ניקח דוגמא עבור $k = 2$**:**
נרצה לחזות את הדירוג של Nicole עבור סרט שהיא לא דירגה, ולבסוף להמליץ לה על סרט בעל הדירוג הגבוה ביותר שחזינו.
נעשה את האבחנה הבאה:

הסרטים אותם Nicole לא דירגה הם: Titanic, Twilight.

הסרטים אותם Nicole דירגה הם: ForestGump, StarWars, Batman.

כעת נחזה את הדירוג שNicole הייתה נותנת עבור הסרטים אותם לא ראתה - על סמך מה שכבר דירגה.

בשביל לחזות את הדירוג של Nicole עבור Titanic, נמצא את הדימיון בינו לבין הסרטים שראתה ודירגה.

– הדמיון בין Titanic ל־ForestGump הוא:

$$\frac{\begin{pmatrix}7\\2\\9\\1\end{pmatrix} \cdot \begin{pmatrix}1\\7\\7\\8\end{pmatrix}}{\left\|\begin{pmatrix}7\\2\\9\\1\end{pmatrix}\right\| \cdot \left\|\begin{pmatrix}1\\7\\7\\8\end{pmatrix}\right\|} = \frac{92}{\sqrt{135} \cdot \sqrt{163}} = 0.62$$

– הדמיון בין Titanic ל־StarWars הוא:

$$\frac{\begin{pmatrix}7\\2\\9\\1\end{pmatrix}\cdot\begin{pmatrix}3\\3\\4\\9\end{pmatrix}}{\left\|\begin{pmatrix}7\\2\\9\\1\end{pmatrix}\right\|\cdot\left\|\begin{pmatrix}3\\3\\4\\9\end{pmatrix}\right\|}=\frac{72}{\sqrt{135}\cdot\sqrt{115}}=0.57$$

– הדמיון בין Titanic ל־Batman הוא:

$$\frac{\begin{pmatrix}7\\2\\9\\1\end{pmatrix}\cdot\begin{pmatrix}2\\6\\4\\8\end{pmatrix}}{\left\|\begin{pmatrix}7\\2\\9\\1\end{pmatrix}\right\|\cdot\left\|\begin{pmatrix}2\\6\\4\\8\end{pmatrix}\right\|}=\frac{70}{\sqrt{135}\cdot\sqrt{120}}=0.55$$

כעת, ידוע כי $k = 2$, ולכן נבחר סט N עם 2 הסרטים שדורגו בצורה הדומה ביותר לTitanic, ובמקרה הזה $N = \{ForestGump,\ StarWars\}$.
הדירוג של Nicole עבור ForestGump ו־StarWars הוא 5 ו6 בהתאמה ולכן חיזוי הדירוג של Nicole עבור Titanic הוא:

$$\frac{0.62\cdot 5+0.57\cdot 6}{0.62+0.57}=\frac{6.52}{1.19}=5.478$$

שימו לב שבקלט הדירוגים הם שלמים אך הדירוגים שחזינו יכולים להיות שברים.

עבור Twilight נחזור על אותו התהליך בדיוק, ונקבל חיזוי של הדירוג: 3.52.

מסקנה: נמליץ על Titanic מכיוון שהדירוג החזוי של Nicole לסרט Titanic הוא הגבוה ביותר מתוך הדירוגים שחזינו עבור הסרטים בN.

שימו לב כי באלגוריתם המלצה לפי תוכן אנו משתמשים בדירוגים מנורמלים ובאלגוריתם המלצה לפי סינון שיתופי אנו משתמשים בדירוגים המקוריים.

2.5 מחלקת RecommenderSystemLoader

מחלקה זו מייצרת מערכת המלצה. במחלקה זו תהיה רק פונקציה סטטית אחת, ואסור להגדיר לה בנאי.

static ptr_type create_rs_from_movies_ le(const std::string& movies_ le_path) noexcept(false)	הפונקציה מקבלת מחרוזת המייצגת נתיב לקובץ מפורמט שיוגדר מטה (2.5.1), ויוצרת מערכת המלצה. הפונקציה מחזירה מצביע בעל בעלות יחידה למערכת ההמלצה (כלומר עליכם להחליף את ptr_type בחתימת הפונקציה בסוג המתאים). על המצביע שיוחזר לא יופעל delete ולכן עליכם להחזיר מצביע מסוג מתאים כך שלא תהיה דליפת זיכרון.

טבלה 4: RecommenderSystemLoader API

2.5.1 מבנה הקובץ המכיל מידע על הסרטים לפי תכונותיהם (קובץ הדוגמה הנ"ל נמצא במודל).

עבור כל סרט, ולכל תכונה, יש ברשותנו score המייצג כמה התכונה תואמת את הסרט. ניתן להניח שהscore הוא מספר אך לא ניתן להניח שהוא מספר בין 1 ל־10. יש לזרוק שגיאה מתאימה (ולשחרר את כל הזיכרון שהוקצה).

ניתן להניח שכל הסרטים הנתונים דורגו עבור כל תכונה נתונה.

ניתן להניח כי אין רווחים בשם הסרט.

ניתן להניח כי לא קיימים שני סרטים בעלי אותו שם.

לצורך פשטות, הקבצים לא יכילו את התכונות המתאימות ואנו נתייחס רק לערכים הרלוונטיים (ראו קובץ דוגמה במודל).

	Scary	Funny	Dramatic	Surprising
Twilight-2008	3	4	5	6
Titanic-1997	7	2	9	1
Batman-2022	2	6	4	8
ForestGump-1994	1	7	7	6
StarWars-1977	3	3	4	9

טבלה 5: מבנה קובץ הקלט עבור יצירת מערכת ההמלצה

2.6 מחלקת RSUsersLoader

מחלקה זו מייצרת משתמשים עם מערכת המלצה מתאימה. במחלקה זו תהיה רק פונקציה סטטית אחת, ואסור להגדיר לה בנאי.

static std::vector<RSUser> create_users_from_ le(const std::string& users_ le_path, ptr_type rs) noexcept(false)	הפונקציה מקבלת מחרוזת המייצגת נתיב לקובץ מפורמט שיוגדר מטה, ומצביע למערכת המלצה מטיפוס שהוחזר מ־ create_rs_from_movies_ le (כלומר ptr_type), ויוצרת משתמשים המקושרים אל מערכת ההמלצה הזו. הפונקציה מחזירה וקטור של כל המשתמשים שהיא יצרה. מערכת ההמלצה לא תימחק עד שאחרון המשתמשים שלה נמחק. שימו לב שאחרי השימוש בפונקציה זו, הבעלות על מערכת ההמלצה עוברת למשתמשים (כלומר לכולם).

טבלה 6: RSUserLoader API

2.6.1 מבנה הקובץ המכיל דירוגים של סרטים לפי שמות משתמשים (קובץ הדוגמה הנ"ל נמצא במודל).

קובץ זה מייצג את הדירוגים של המשתמשים עבור סרטים שהם ראו (בהנחה כי בכל סיום של צפייה בסרט הם דירגו את הסרט לפי מספר מ1 עד 10 (או ערך ריק (NA)) אם הם לא ראו את הסרט).

	Twilight-2008	Titanic-1997	ForestGump-1994	Batman-2022	StarWars-1977
So a	4	NA	8	NA	NA
Michael	NA	8	4	NA	9
Nicole	NA	NA	5	2	6
Arik	NA	8	NA	3	NA

טבלה 7: מבנה קובץ הקלט עבור יצירת המשתמשים

ניתן להניח כי:

1. לא קיימים שני משתמשים בעלי אותו שם.
2. לא קיימים שני סרטים בעלי אותו שם שיצאו באותה שנה באותה מערכת המלצה.
3. כל משתמש דירג לפחות סרט אחד.
4. כל משתמש לא דירג לפחות סרט אחד.
5. כמו בקובץ הקודם ניתן להניח שעבור כל איבר בטבלה מופיע מספר כלשהו או NA, אך לא ניתן להניח שמדובר במספר שלם וחיובי. אם המספר לא בין 1 ל־10 יש לזרוק שגיאה מתאימה (ולשחרר את כל הזיכרון שהוקצה).

שימו לב: סדר הסרטים בעמודות בקובץ זה לא בהכרח תואם את סדר השורות בקובץ לעיל.

עבור שתי המחלקות הנ"ל, שימו לב כי:

שתי המחלקות (RecommenderSystemLoader, RSUserLoader) הן מחלקות סטטיות ולכן אין להן בנאי (כלומר אי אפשר ליצור מהן אובייקט) והן לא שומרת אובייקטים.

ייתכן שהפונקציות הסטטיות ייקראו יותר מפעם אחת בהרצה כלשהי ובכל פעם הפונקציה תידרש להחזיר אובייקט חדש.

הניחו כי מאגר המידע שאיתו התוכנה תעבוד נמצא בשרת רחוק המצריך פעולות תקשורת מרובות על מנת לגשת אליו. לכן, עליכם להימנע מגישות I/O (Input / Output) מיותרות, ולגשת לקבצים **פעם אחת בלבד** על מנת לקרוא את המידע מתוך הקבצים.

מערכת ההמלצה היא מערכת כבדה ולכן אין להעתיק אותה עבור משתמשים שונים. נהלו את הזיכרון בצורה חכמה כך שמערכת ההמלצה לא תימחק עד שאחרון המשתמשים שלה נמחק. שימו לב שאחרי השימוש בפונקציה של המחלקה, הבעלות עוברת למשתמשים (לכולם).

ניתן להניח שלא נשתמש ב־ptr_type שמתקבל מ־create_rs_from_movies_le אחרי שהבעלות על המצביע הועברה לפחות למשתמש אחד.

כאשר מדברים על שגיאה מתאימה הכוונה לאחת השגיאות שנראו בכיתה או המופיעות ב־ https://en.cppreference.com/w/cpp/error/exception. יש יותר משגיאה אחת מתאימה ואנחנו נקבל כל שגיאה מתאימה.

הבהרות:

אפשר לשכפל קוד באופן מינימלי בין קובצי הLoaders-

ניתן ואף מומלץ להוסיף include נוספים משלכם לקבצים בתרגיל, אך אין לשנות את ה-includes שניתנו לכם בשלד.

ניתן להגדיר קומפרטור פומבי עבור מימוש מבנה הנתונים שלכם, בדומה ל sp_movie_equal-שמומש עבורכם. שימו לב - עבור unordered_map אתם נדרשים לספק פונקציית גיבוב ופונקציית שוויון, אך עבור map אתם צריכים לספק Comparator.

כאשר בוחרים k סרטים בפונקציות הנדרשות לכך, אם יש צורך בשבירת שיוויון אתם יכולים לשבור אותו איך שאתם רואים לנכון. לא נבדוק אתכם על זה.

2.7 הערות

1. **שימו לב שהקוד שאתם מגישים אינו מכיל** main
2. חלק מה־API אינו כולל את החתימות המפורשות. עליכם להשלים את החתימה בצורה מתאימה, בהתאם למה שראיתם בהרצאות ובתרגולים כולל מספר פונקציות לגרסא של const ו־non-const אם זה נראה לכם מתאים לפונקציה.
3. מותר להוסיף פונקציות פרטיות אך אין להוסיף פונקציות פומביות שאינן מופיעות ב־API הנתון (פונקציות פומביות שלא הוגדרו ב־API יכולות להשפיע על הטסטים).
4. לצורך קריאת הנתונים מהקבצים, הנכם רשאים (**ומומלץ**) להעזר במחלקה istringstream.
5. על מנת לבדוק את הקוד שלכם, תוכלו למצוא במודל שני קבצי קלט לדוגמא המכילים את הדוגמא שראיתם במסמך זה בפורמט הנכון.

3 נהלי הגשה

קראו בקפידה את הוראות תרגיל זה ואת ההנחיות להגשת תרגילים שבאתר הקורס. כמו כן, זכרו כי התרגילים מוגשים ביחידים. אנו רואים העתקות בחומרה רבה!

עליכם להשתמש במבני הנתונים בספריית STL **וכן מומלץ להשתמש באלגוריתמים המוצעים בספריה.** מטרת התרגיל היא **שימוש ניכר במבני נתונים של** STL **וכן באלגוריתמים** ־ שימו לב כי קוד נכון ויעיל הוא קוד המשתמש במבנים הנכונים והיעילים ביותר למשימה, ומכאן גם קוד המשתמש באלגוריתמים שהספרייה מציעה. בנוסף, שימו לב שמימוש נכון של התרגיל כולל שימוש במצביעים חכמים.

שימו לב להערות ולהנחות שניתנו לכם, ובעיקר לאלו המסומנות באדום!

ההגשה נעשית דרך הגיט, שימו לב שאתם מעלים רק את הקבצים המתאימים ־

Movie.h, Movie.cpp

RecommenderSystemLoader.h, RecommenderSystemLoader.cpp

RSUsersLoader.h, RSUsersLoader.cpp

RecommenderSystem.h, RecommenderSystem.cpp

RSUser.h, RSUser.cpp

אנא וודאו כי התרגיל שלכם עובר את ה־Pre-submission Script **ללא שגיאות או אזהרות**. קובץ ה־Pre-submission Script זמין בנתיב:

```
~proglab/presubmit/ex5/run <path/to/submission.tar>
```

שימו לב כי בקבצי התרגיל <u>אתם לא מגישים פונקציית main</u>, אלא **רק את המחלקות**, אך עליכם לבדוק כי התוכנית מתקמפלת כאשר אתם מכניסים פונקציית main המדמה הרצה של הספרייה כאשר ניתן להשתמש בקובצי הקלט שנתונים לכם, ולפי הפקודה הבאה:

```
g++ -Wall -Wvla -Wextra -Werror -g -std=c++14 <code  les> -o prog
```

שימו לב שבדיקת ה־coding style נעשית כחלק מה־presubmit.

במידה והשתמשתם בהקצאות זיכרון, עליכם לדאוג לניהול ושחרור הזיכרון ללא דליפות, כולל מימוש נכון של חוק ה־3 במקומות בהם הוא נדרש (שימו לב כי במקרים מסוימים ראיתם שהוא לא נדרש). תוכלו להיעזר ב־valgrind כדי לבדוק האם בתרגילכם יש דליפות זיכרון. עליכם להריץ את הפקודה:

```
valgrind --leak-check=full <Command to Debug>
```

4 נספח ־ הגדרות

1. **נורמה**

נורמה היא פונקציה ממשית המוגדרת על מרחב וקטורי, ומתאימה לכל וקטור ערך ממשי, באופן שמקיימות האקסיומות הבאות:

(א) חיוביות:

$$||x|| = 0 \Rightarrow x = 0 \ \wedge \ ||x|| \geq 0$$

(ב) הומוגניות:

$$||\lambda x|| = ||\lambda|| \cdot ||x||$$

(ג) אי שיוויון המשולש:

$$||x|| + ||y|| \geq ||x + y||$$

2. **מכפלה סקלרית**

מכפלה סקלרית היא פעולה על שני וקטורים מהמרחב האוקלידי R^n, שמחזירה סקלר. לדוגמא, יהיו $\alpha, \beta \in R^n$, המוגדרים באופן הבא:

$$\alpha = (\alpha_1, ..., \alpha_n)$$

$$\beta = (\beta_1, ..., \beta_n)$$

אזי המכפלה הסקלרית בין α, β היא מוגדרת ומסומנת:

$$\alpha \cdot \beta = \alpha_1 \cdot \beta_1 + ... + \alpha_n \cdot \beta_n$$

3. **הנורמה הסטנדרטית במרחב האוקלידי:**

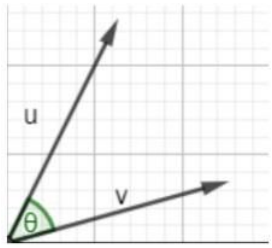
$$||x|| = \sqrt{x_1^2 + ... + x_n^2}$$

זוהי הנורמה בה נשתמש לאורך כל התרגיל.

4. **זווית בין וקטורים**

בהינתן וקטורים u, v, נוכל לחשב את הזווית ϑ בניהם בצורה הבאה:

$$\vartheta = cos^{-1}\left(\frac{u \cdot v}{||u|| \cdot ||v||}\right)$$



איור 1: זווית בין וקטורים

הערה: הפונקציה ההופכית של cos מוגדרת בתחום $[-1, 1]$ ומונוטונית יורדת בתחום זה, משמע ככל ש־$\frac{u \cdot v}{||u|| \cdot ||v||}$ מתקרב ל־1, הזווית בין u וv קטנה, וככל ש$\frac{u \cdot v}{||u|| \cdot ||v||}$ מתקרב ל־(1 _), הזווית גדלה.

מסקנה: נוכל למדוד דמיון בין ווקטורים (דמיון הכיוונים) לפי חישוב הזווית $\vartheta = \frac{u \cdot v}{||u|| \cdot ||v||}$. ככל שערך זה יותר גבוה, הווקטורים u ו־v דומים יותר.

שימו לב כי אם $\frac{u \cdot v}{||u|| \cdot ||v||} = 1$, למעשה $u = v$.

בהצלחה!