האוניברסיטה העברית בירושלים

בית הספר להנדסה ולמדעי המחשב ע"ש רחל וסלים בנין

סדנת תכנות בשפות c & c++
67315

תרגיל רשות 5 - תרגיל מסכם בשפת סי++ עם הקלות

על מנת להקל - ניתנו לכם בקבצים המצורפים חלקים נרחבים מהפתרון

תאריך אחרון להגשה: 31.7.2024 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.ן החדשה ראו שאתם.ן מתכנתי C++ נהדרים.ות ולכן ביקשו מכם.ן לממש פרויקט בעצמכם.ן. מטרת הפרויקט היא לכתוב ספריה שתשמש את חברות הזרמת המדיה (Streaming) הגדולות. הספרייה היא כלי ניהול של מערכות המלצת סרטים ומשתמשים, שתאפשר לחברות הזרמת המדיה להמליץ ללקוחותיהן על הסרטים המתאימים ביותר לכל משתמש.
שימו לב, לפי שאתם.ן ניגשים.ות למבנה הספרייה:

- כדי להבין מהו סרט במערכת ההמלצה וכיצד סרט מיוצג, עברו בקפידה על מבנה הקובץ המכיל מידע על הסרטים לפי תכונותיהם.
- למערכת ההמלצה שלנו 2 אלגוריתמי המלצה: המלצה לפי תוכן והמלצה לפי סינון שיתופי. עברו בקפידה עליהם ועל הדוגמות שסופקו.

עליכם.ן לממש את הספרייה כפי שמוגדר להלן.

2 בניית המשתמשים ומערכת ההמלצות

2.1 מבנה הספרייה

הספרייה מורכבת מחמש מחלקות:
1. Movie - מחלקה המייצגת סרט במערכת.
2. User - מחלקה המייצגת משתמש של המערכת. למשתמש מספר פונקציות כדי לקבל המלצה מהמערכת, המבוססות על האלגוריתמים שממומשים ב- RecommendationSystem.
3. RecommendationSystem - מחלקה המספקת המלצות צפיה עבור משתמש מסוים (מערכת ההמלצה). המחלקה כוללת מספר אלגוריתמים בעזרתם משתמש יכול לקבל המלצה על סרט.
4. RecommendationSystemLoader - מחלקה האחראית לייצור מערכת ההמלצה.
5. UsersLoader - מחלקה האחראית לייצור המשתמשים.

במסגרת ההקלות שניתנו בתרגיל, המימוש של מחלקות RecommendationSystemLoader ו- UsersLoader ניתן לכם.ן, וכל שעליכם.ן לעשות במחלקות אלו הוא להשלים את סוג ה-ptr המתאים במקום המתאים.
עם זאת - החלק שקובץ זה שמדבר על המחלקות הממומשות והקוד שקיבלתם הינו חלק אינטגרלי מהוראות התרגיל.

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

המתאימים בקבצי ה-cpp שהוענקו לכם.ן בגרסה המקוצרת של התרגיל.

שימו ♥: עברו על הקוד הנתון לכם.ן **לפני** תחילת כתיבת הקוד.

שימו ♥: קראו את כל המסמך **לפני** תחילת כתיבת הקוד, על מנת להבין את מבנה המערכת ולמנוע כפל קוד.

2.1.1 הסבר קצר לגבי הפונקציונליות של המערכת

זוהי פיסקה שנועדה להסביר ולסדר מעט את הרעיון שמאחורי המבנה של התרגיל, במידה ויש סתירה בין מה שכתוב בפיסקה זו לבין מה שכתוב בשאר המסמך אין להתחשב במה שכתוב כאן.

תחילת העבודה של הלקוח עם המערכת הוא לפתוח מערכת המלצה חדשה ולמלא אותה ברשימה של סרטים באמצעות המחלקה RecommendationSystemLoader , לאחר מכן, הוא משתמש במחלקה UsersLoader כדי לייצר רשימה של יוזרים משתמשים במערכת ההמלצה (RecommendationSystem) אותה הוא יצר לפני כן. לאחר מכן, בכל פעם שאחד המשתמשים רוצה לקבל המלצה לסרט מהמערכת הוא יכול לבקש המלצה דרך אחת הפונקציות הממליצות שנמצאות במחלקה User (פונקציות אלו משתמשות באלגוריתמים של מערכת ההמלצה). כמו כן, במידה ואחד המשתמשים ראה סרט חדש (שקיים או שלא קיים במערכת כבר) והוא רוצה לתת לו דירוג הוא יכול להשתמש בפונקציה add_movie_to_rs ופונקציה זו תוסיף את הדירוג שלו אם הסרט כבר קיים במערכת ואם לא אז תוסיף את הסרט למערכת ואת הדירוג של אותו לקוח לאחר מכן.

2.2 מחלקת Movie

נוסף על ה־API המפורט מטה שעליכם.ן לממש, עליכם.ן להגדיר בקובץ ה־h של המחלקה typedef הנקרא sp_movie. ה־typedef יהיה של מצביע מטיפוס Movie. בחרו סוג 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 אותן (אי מימושן עלול לגרום לכישלון בטסטים.)

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 (ה-<> לא אמורים להופיע בהדפסה)
operator <	אופרטור השוואה < ביחס לשנת ההוצאה של הסרט. אם הסרטים יצאו באותה השנה, האופרטור מחזיר true אם שם הסרט השמאלי קטן משם הסרט הימני לפי סדר לקסיקוגרפי. לדוגמה: עבור הסרטים Twilight-2008, Titanic-1997, Wanted-2008, מתקיימים היחסים הבאים: Titanic-1997 < Twilight-2008 מכיוון ששנת ההוצאה של "Titanic" קטנה משנת ההוצאה של "Twilight". כמו כן, Twilight-2008 < Wanted-2008 מכיוון ששניהם יצאו באותה שנה אך "Twilight" קטן יותר (לפי סדר לקסיקוגרפי) מ-"Wanted". שימו לב **לתיעוד הפונקציה בקובץ השלד**, בפרט לכמה פרמטרים היא אמורה לקבל.
get_name	הפונקציה מחזירה את שם הסרט
get_year	הפונקציה מחזירה את שנת ההוצאה של הסרט

טבלה 1: Movie API

<u>דגשים והנחות:</u>

- ניתן להניח את תקינות הקלט.
- שני סרטים a b, נחשבים זהים אם כל אחד מהם לא קטן מהשני, כלומר:
$$!(a < b) \wedge !(b < a)$$
- חלק מה-API אינו כולל את החתימות המפורשות. עליכם.ן להשלים את החתימה בצורה המתאימה בהתאם למה שראיתם.ן בהרצאות ובתרגולים.

בקובץ השלד שניתן לכם.ן עבור המחלקה, יש מספר הגדרות חשובות. אסור למחוק אותן ועליכם.ן להשתמש בהן במהלך כתיבת התרגיל[1].

```
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 (sp_movie_hash) ו-(sp_movie_equal), קיים מימוש בקובץ .Movie.cpp אין לשנות או להזיז את המימוש - שינוי או הזזה עלול לגרום לכישלון בטסטים.

2.3 מחלקת User

מחלקה זו מייצגת משתמש אחד.

בשלד של מחלקה זו מוגדר לכם.ן ה־typedef הבא, ועליכם.ן להשתמש בו במהלך מימוש ה־API:

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

Constructor	הבנאי מקבל מחרוזת המייצגת את שם המשתמש, את הדירוגים שלו לסרטים שכבר ראה ומצביע למערכת המלצה. החתימה המדויקת של הבנאי נתונה לבחירתכם (שימו לב לטיפוסי המשתנים בחתימה). ניתן להניח כי לא קיימים שני משתמשים בעלי אותו שם. ניתן להניח שהשם איננו הסטרינג הריק.
get_name()	הפונקציה מחזירה את שם המשתמש
void add_movie_to_user(const std::string &name, int year, const std::vector<double>& features, double rate)	הפונקציה מקבלת מחרוזת המייצגת את שם הסרט, מספר המייצג את שנת ההוצאה של הסרט, את ערכי התכונות השונות של הסרט ואת דירוג המשתמש לסרט זה. היא מוסיפה אותו למאגר של מערכת ההמלצה ושומרת את ערכי התכונות שלו (שימו לב כי שינוי זה משפיע על כל המשתמשים של מערכת ההמלצה שכן למערכת נוסף כעת סרט חדש שניתן להציע למשתמשים). כמו כן, היא מוסיפה את הסרט לרשימת הדירוגים של המשתמש. אם הסרט כבר קיים, יש לדרוס את הדירוג הנוכחי שלו. ניתן להניח תקינות הקלט.
sp_movie get_rs_recommendation_by_content () const	הפונקציה מחזירה מצביע לסרט המומלץ לפי אלגוריתם המלצה לפי תוכן.

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

double get_rs_prediction_score_for_movie(const std::string& name, int year, int k) const	הפונקציה מקבלת מחרוזת המייצגת את שם הסרט, 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־RecommendationSystem.
sp_movie get_rs_recommendation_by_cf(int k) const	הפונקציה מקבלת מספר שלם וחיובי (הפרמטר באלגוריתם הסינון השיתופי) המייצג את מספר הסרטים הדומים ביותר לכל סרט (ומדורגים על ידי המשתמש) עליהם נתבסס בחיזוי. הפונקציה מחזירה מצביע לסרט עליו נמליץ למשתמש לפי שיטת סינון שיתופי כפי שיוסבר בהמשך. ניתן להניח כי הפרמטר k שלם וחיובי, וקטן ממספר הסרטים שדורגו על ידי המשתמש.
operator <<	אופרטור שמעביר לostream את שם המשתמש בפורמט: :NAME>\n“ ”name> ולאחר מכן את כל הסרטים במערכת ההמלצה (גם את אלו שהוא לא ראה) בצורה ממוינת (לפי האופרטור >) על פי פורמט ההדפסה של סרט. לאחר הדפסת כל הסרטים יש לרדת שורה (endl) (בנוסף לירידת השורה המתבצעת בפונקציית הדפסת הסרטים של המערכת).
get_ranks()	הפונקציה מחזירה את הדירוגים של המשתמש עצמו (כלומר מחזירה אובייקט מסוג rank_map)

טבלה 2: User API

<u>דגשים והנחות:</u>

- המחלקות RecommendationSystemLoader, UsersLoader מאתחלות את המשתמשים ואת מערכת ההמלצה מתוך הקבצים המתאימים. המטרה של הפונקציה add_movie_to_rs של מחלקת User 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. ניתן להניח שהארגומנט rate הנו מספר שלם בין 1 ל-10.

- ניתן להניח שהוספת סרטים היא עקבית, כלומר אם במהלך יצירת מערכת ההמלצה היו 5 תכונות לסרטים, כאשר נוסיף סרט חדש יהיה גם לו 5 תכונות.

2.4 מחלקת RecommendationSystem

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־(2.4.2) (2.4.1 API,
נגדיר את ה־API של המחלקה:

Constructor	בנאי שאינו מקבל פרמטרים
sp_movie recommend_by_content (const User& user_rankings)	הפונקציה מחזירה מצביע חכם לסרט המומלץ לפי אלגוריתם המלצה לפי תוכן(2.4.1). ניתן להניח שהפונקציה תיקרא **רק עבור משתמשים שדירגו יותר מסרט אחד עם דירוגים שונים.**

double predict_movie_score(const User& user_rankings, const sp_movie &movie, int k)	הפונקציה מקבלת 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 (2.4.2). **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 User& user_rankings, int k)	הפונקציה מקבלת 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. הפונקציה 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_to_rs(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 מחזירה מצביע לסרט. **ניתן להניח שהסרט לא נמצא במערכת, שם הסרט הוא לא סטרינג ריק ושמספר התכונות של הסרט זהה למספר התכונות של סרטים שקיימים במערכת.** דרישת היעילות של פונקציה זו היא: $O(\log log\ (n))$ כאשר n הוא מספר הסרטים במערכת.
get_movie	הפונקציה מחזירה 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: $O(\log log\ (n))$ 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: RecommendationSystem API

דגשים:

- עליכם.ן לממש את המחלקה בצורה יעילה, ובפרט עליכם.ן לחשוב על מבנה נתונים מתאים מתוך STL שיאפשר מימוש העומד ביעילות הנדרשת. כדי שמבנה הנתונים שבחרתם.ן יעמוד בדרישות, ניתן ואף מומלץ לקרוא ולהיעזר בקוד ובמבנה הנתונים שסופק לכם ב־User לשם השראה, ולהשתמש בו עם שינויים מתאימים.
- ניתן להניח כי פונקציות ההמלצה יקראו רק אם במערכת ההמלצה יש סרט שהמשתמש טרם ראה.
- 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.
- ניתן להניח כי לא יהיו סרטים אצל היוזר שלא נמצאים במערכת.

2.4.1 המלצה לפי תוכן

יש לעבור על נספח ההגדרות לפני קריאת הסבר זה

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

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

שלב 2: ניצור וקטור העדפה של תכונות (כלומר, גודל הוקטור או כגודל מספר התכונות שיש לסרט במערכת ההמלצה) למשתמש x , המבוסס על הדירוגים שלו לסרטים, ביחד עם תכונותיהם של אותם סרטים.

שימו לב כי הקאורדינטה ה־i בוקטור התוצאה בסיום השלב הנוכחי מייצגת 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.

דוגמה:

נרצה להמליץ לSofia על סרט לפי שיטת המלצה לפי תוכן.

שלב 1

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, כאשר נתון שווקטור התכונות של Titanic הנו:

Titanic (1997)	7	2	9	1

2. הדירוג המנורמל של Sofia ל-ForestGump הוא 2, כאשר נתון שווקטור התכונות של ForestGump הנו:

ForestGump (1994)	1	7	7	6

נקבל כי בסה"כ וקטור ההעדפות של סופיה הוא:

$$-2\cdot(7\ 2\ 9\ 1) + 2\cdot(1\ 7\ 7\ 6) = (-12\ 10\ -4\ 10)$$

אינטואיציה: סופיה אוהבת מאוד סרטים מפתיעים ומצחיקים, ובאותה מידה, לא אוהבת סרטים דרמטיים ומאוד לא אוהבת סרטים מפחידים.

שלב 3

חישוב הדמיון בין וקטור ההעדפות של סופיה לוקטורי התכונות של הסרטים ש־ Sofiaלא דירגה – StarWars, Twilight ו-Batman.

1. וקטור התכונות של Twilight :

Twilight (2008)	3	4	6	5

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

$$\frac{(-12\ 10\ -4\ 10)\bullet(3\ 4\ 6\ 5)}{\|(-12\ 10\ -4\ 10)\|\bullet\|(3\ 4\ 6\ 5)\|} = \frac{30}{\sqrt{360}\bullet\sqrt{86}} = 0.17$$

Batman (2022)	2	6	4	8

2. וקטור התכונות של Batman :

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

$$\frac{(-12\ 10\ -4\ 10)\bullet(2\ 6\ 4\ 8)}{\|(-12\ 10\ -4\ 10)\|\bullet\|(2\ 6\ 4\ 8)\|} = \frac{100}{\sqrt{360}\bullet\sqrt{120}} = 0.48$$

StarWars (1977)	3	3	4	9

3. וקטור התכונות של StarWars :

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

$$\frac{(-12\ 10\ -4\ 10)\bullet(3\ 3\ 4\ 9)}{\|(-12\ 10\ -4\ 10)\|\bullet\|(3\ 3\ 4\ 9)\|} = \frac{68}{\sqrt{360}\bullet\sqrt{115}} = 0.33$$

מסקנה: נמליץ ל-Sofia על הסרט Batman מכיוון שהתכונות שלו הכי דומות להעדפות של Sofia

2.4.2 המלצה לפי סינון שיתופי ((Collaborative filtering

יש לעבור על נספח ההגדרות לפני קריאת הסבר זה

רעיון כללי:

נרצה לתת המלצה למשתמש על סרט שהוא לא ראה, **בהסתמך על הסרטים שהוא ראה שהם הדומים ביותר לסרט שהוא טרם ראה.**

כלומר, עבור סרט m, נמצא סט $N = \{N_1, ..., N_k\}$ המכיל k סרטים מבין הסרטים שהמשתמש דירג, שהכי דומים לסרט m. לפי הסט שמצאנו, נחזה את הדירוג של משתמש x לסרט m. נעשה זאת באמצעות פונקציה הלוקחת את הדירוגים של המשתמש עבור אותם $N_1, ..., N_k$, הסרטים בסט, ומחשבת את הדימיון (הזווית) בין N_j לכל $j\in[1, k]$, לסרט m.

על מנת לחזות את הדירוג של משתמש x עבור סרט m, (שהמשתמש עוד לא ראה!) נפעל בצורה הבאה:

שלב 1:

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

<u>שלב 2:</u>

נחזה את הדירוג של משתמש x לסרט הm באופן הבא:

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

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

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.**

- ניתן להניח שלא ניתן לכם מקרה בו תצטרכו לחלק ב-0 כאשר מנסים לחזות ציון של סרט עבור משתמש כלשהו

נדגים עבור $k := 2$

נרצה לחזות כמה תדרג Nicole את כל הסרטים שהיא טרם ראתה ודירגה, ולבסוף להמליץ לה על סרט בעל הדירוג הגבוה ביותר לפי התחזית שלנו.

הסרטים אותם Nicole לא דירגה הם: Titanic. Twilight,

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

בשביל לחזות את הדירוג של Nicole עבור Titanic, נמצא כמה דומה Titanic לסרטים שהיא כן ראתה ודירגה.

הדמיון של Titanic ושל ForestGump הנו:

$$\frac{(7\ 2\ 9\ 1)\bullet(1\ 7\ 7\ 8)}{\|(7\ 2\ 9\ 1)\|\bullet\|(1\ 7\ 7\ 8)\|} = \frac{92}{\sqrt{135}\bullet\sqrt{163}} = 0.62$$

הדמיון של Titanic ושל StarWars הנו:

$$\frac{(7\ 2\ 9\ 1)\bullet(3\ 3\ 4\ 9)}{\|(7\ 2\ 9\ 1)\|\bullet\|(3\ 3\ 4\ 9)\|} = \frac{72}{\sqrt{135}\bullet\sqrt{115}} = 0.57$$

הדמיון של Titanic ושל Batman הנו:

$$\frac{(7\ 2\ 9\ 1)\bullet(2\ 6\ 4\ 8)}{\|(7\ 2\ 9\ 1)\|\bullet\|(2\ 6\ 4\ 8)\|} = \frac{70}{\sqrt{135}\bullet\sqrt{120}} = 0.55$$

כעת, עבור $k = 2$ נבחר את N להיות $N = \{ForestGump, StarWars\}$ שכן אלו הם שני הסרטים שהכי דומים לסרט Titanic מבין הסרטים ש-Nicole ראתה ודירגה.

הדירוג של Nicole עבור ForestGump ו־StarWars הוא 5 ו-6 בהתאמה ולכן חיזוי הדירוג של Nicole עבור Titanic לפי האלגוריתם שסופק הוא:

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

***שימו לב** שבקלט הדירוגים הם בשלמים אך הדירוגים שחזינו יכולים להיות שבריים.*

עבור הסרט Twilight נחזור בדיוק על אותו התהליך, ונקבל כי לפי התחזית שלנו, Nicoleתדרג את הסרט: 3.52.

***מסקנה:** נמליץ ל-Nicole לראות Titanic מכיוון שלפי האלגוריתם שלנו, הניקוד ש-Nicole תעניק ל-Titanicהוא הגבוה ביותר מבין הסרטים שהיא טרם ראתה.*

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

2.5 מחלקת RecommendationSystemLoader

מימוש של מחלקה זו ניתן לכם במסגרת ההקלות בתרגיל.

כל שעליכם לעשות הוא להשלים את ה- ptr_type בחתימת הפונקציה בסוג המתאים (בקבצי ה-h וה-cpp).

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

static ptr_type create_rs_from_movies(const std::string& movies_ le_path) noexcept(false)	הפונקציה מקבלת מחרוזת המייצגת נתיב לקובץ לפי הפורמט המוגדר 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: RecommendationSystemLoader API

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

לכל סרט, ולכל תכונה, יש ברשותנו score המייצג כמה התכונה תואמת את הסרט.

<u>דגשים והנחות:</u>

- ניתן להניח שכל הסרטים הנתונים דורגו עבור כל תכונה נתונה.
- ניתן להניח כי לא קיימים שני סרטים בעלי אותו שם.
- לשם פשטות, הקבצים לא יכילו את שמות התכונות (Scary, Funny, etc..) אלא רק את הערכים הרלוונטיים (ראו קובץ דוגמה במודל).
- לא ניתן להניח את מספר התכונות שיש לסרטים שתקבלו (כלומר לא ניתן להניח שבקובץ שתקבלו לסרט יש 4 תכונות כמו בקובץ מטה), אך כאמור, כן ניתן להניח שהוספת סרטים היא עקבית, כלומר אם במהלך יצירת מערכת ההמלצה לסרטים היו 5 תכונות, כאשר נוסיף סרט חדש גם לו יהיו 5 תכונות.

2.6 מחלקת UsersLoader

מימוש של מחלקה זו ניתן לכם במסגרת ההקלות בתרגיל.

כל שעליכם לעשות הוא להשלים את ה- ptr_type המתאים <u>במקומות</u> המתאימים, (בקבצי ה-h וה-cpp).

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

static std::vector<User> create_users(const std::string& users_file_path, ptr_type rs) noexcept(false)	הפונקציה מקבלת מחרוזת המייצגת נתיב לקובץ מפורמט המוגדר מטה, ומצביע למערכת המלצה מטיפוס שהוחזר מ־ create_rs_from_movies (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: UsersLoader 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
Sofia	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

<u>דגשים והנחות:</u>

- לא קיימים שני משתמשים בעלי אותו שם.
- לא קיימים שני סרטים בעלי אותו שם שיצאו באותה שנה באותה מערכת המלצה.
- כל משתמש דירג לפחות סרט אחד.
- כל משתמש לא דירג לפחות סרט אחד.
- **בחנו את הקוד הניתן לכם.ן כדי להבין איך NA נראה בפועל במערכת.**

<u>הבהרות נוספות:</u>

- ניתן ואף מומלץ להוסיף include נוספים משלכם.ן לקבצים בתרגיל, אך אין לשנות את ה-includes שניתנו לכם.ן בשלד.
- ניתן להגדיר Comparator פומבי עבור מימוש מבנה הנתונים שלכם.ן, בדומה ל-sp_movie_equal שמומש עבורכם.ן. שימו לב - עבור unordered_map אתם.ן נדרשים.ות לספק פונקציית גיבוב ופונקציית שוויון, אך עבור map אתם.ן צריכים.ות לספק Comparator.

- בדומה למה שראינו בתרגול (ל-set שהוגדר עם פונקציית comparator, צריך לספק ב-constructor את הפונקציה), כאשר נרצה ליצור אובייקט unordered_map, נצטרך לספק ל- constructorאת הגודל ההתחלתי (והמינימלי) של מספר התאים אליהם יגובבו האיברים לפי ערך ה-hash שלהם, לאחר מכן את פונקציית ה-hash ולבסוף את פונקציית ה-equal.
להרחבה https://cplusplus.com/reference/unordered_map/unordered_map/unordered_map

הערות

1. **שימו לב שהקוד שאתם.ן מגישים.ות אינו מכיל main**
2. חלק מה־API אינו כולל את החתימות המפורשות. עליכם.ן להשלים את החתימה בצורה מתאימה, בהתאם למה שראיתם.ן בהרצאות ובתרגולים כולל מספר פונקציות לגרסה של const ו-non-const אם זה נראה לכם.ן 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. על מנת לבדוק את הקוד שלכם.ן, תוכלו למצוא במודל שני קבצי קלט לדוגמא המכילים את הדוגמא שראיתם.ן 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.ן 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 ה-git, שימו לב שאתם מעלים רק את הקבצים המתאימים:
  1. Movie.cpp
  2. Movie.h
  3. User.cpp
  4. User.h
  5. RecommendationSystem.cpp
  6. RecommendationSystem.h
  7. RecommendationSystemLoader.cpp
  8. RecommendationSystemLoader.h
  9. UsersLoader.cpp
  10. UsersLoader.h

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

- קובץ ה־Script Pre-submission זמין בנתיב:

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

- שימו לב כי בקבצי התרגיל אתם לא מגישים פונקציית main, 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 (שימו לב להשתמש ב-C++14):

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

- שימו לב שבדיקת ה־style coding 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.) לשם כך 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 שמקיים את האקסיומות הבאות:

1. חיוביות:

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

2. הומוגניות:

$$\lambda \in R,\ ||\lambda x|| = ||\lambda|| * ||x||$$

3. 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:

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

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

המכפלה הסקלרית בין α β, היא מוגדרת ומסומנת:

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

3. הנורמה הסטנדרטית במרחב האוקלידי:

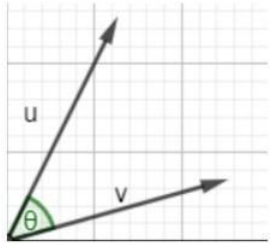
$$\|x\| = \sqrt{x_1^2 + \ldots + x_n^2}$$

זוהי הנורמה בה נשתמש לאורך כל התרגיל.

4. זווית בין וקטורים

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

$$\theta = \left(\frac{u \cdot v}{\|u\| \cdot \|v\|}\right)$$



איור 1: זווית בין וקטורים

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

מסקנה: נוכל למדוד דמיון בן ווקטורים (דמיון הכיוונים) לפי חישוב הזווית $\theta = \frac{u \bullet v}{\|u\| \cdot \|v\|}$. ככל שערך זה יותר גבוה, הווקטורים דומים יותר. שימו

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

בהצלחה!