תכנות מתקדם – 150024

תרגיל בית מספר 5

קבצים בינארים

שים/י לב:

א. הקפד על קְרִיאוּת התכנית ועל עימוד (Indentation).
ב. הקפד לבצע בדיוק את הנדרש בכל שאלה.
ג. בכל אחת מהשאלות יש להגדיר כל מחלקה ב 2 קבצים נפרדים. קובץ H וקובץ CPP.
ד. בכל אחת מהשאלות יש להגדיר פונקציות עזר במידת הצורך עבור קריאות התכנית.
ה. יש להגיש את התרגיל על פי ההנחיות להגשת תרגילים (המופיע באתר הקורס) וביניהם:
ו. השתמש/בשמות משמעותיים עבור המשתנים.
ז. יש לתעד את התכנית גם עבור פונקציות אותם הנך מגדיר וכן על תנאים ולולאות וקטעי קוד מורכבים, ובנוסף, **דוגמת הרצה לכל תכנית בסוף הקובץ**!
ח. **הגשה יחידנית - אין להגיש בזוגות.**

הערה חשובה: לכל תרגיל בית מוגדר שבוע אחד בלבד להגשה, אלא אם כן קיבלת הוראה אחרת מהמרצה שלך. תיבות ההגשה הפתוחות לא מהוות היתר להגשה באיחור.

גישה אקראית לקובץ בינארי (Random Access Binary Files).

בתרגיל זה עליך לנהל קובץ בינארי בו שמורים נתוני רישום סטודנטים לקורסים שונים.

שלב 1:

מחלקת Student

א. הגדר מחלקה בשם **Student** לייצוג סטודנט.
השדות במחלקה יהיו:
- id - **מספר מזהה** (int) – מספר בין 1 ל- 100.
- firstName - **שם פרטי** (מערך של 20 תוים (ללא הקצאה דינאמית או string))
- lastName - **שם משפחה** (מערך של 20 תוים (ללא הקצאה דינאמית או string))
- courses - **קורסים** – מערך בוליאני עבור 5 קורסים.

נניח כי ניתן להירשם בכל סמסטר מקסימום ל- 5 קורסים. המערך מייצג את כל 5 הקורסים הקיימים. עבור כל קורס, במידה והסטודנט נרשם לקורס ישמר ערך true ואחרת ישמר הערך false.

ניתן להניח שכל ערכי השדות שיקלטו יהיו תקינים. **אין צורך לבדוק תקינות קלט**.

ב. הוסף למחלקה את הבנאים הבאים:
- **default constructor** ערכי ברירת המחדל של המחלקה יהיו:
  - id = 0
  - firstName=""
  - lastName=""
  - courses={false,false,false,false,false}

- **assignment constructor** בנאי המקבל מספר זהות, שם פרטי, שם משפחה ומאתחל את השדות בהתאם.
בשלב זה הסטודנט עדיין אינו רשום לאף קורס ולכן יש לאתחל את המערך courses={false,false,false,false,false}

ג. הוסף למחלקה Student **רק** את המתודות הבאות:

- getId() – מתודה המחזירה את מספר הזהות של הסטודנט.

- operator[] – מתודה בוליאנית המקבלת מספר קורס ומחזירה true אם הסטודנט רשום לקורס ו-false אם אינו רשום לקורס.
<u>הערה</u>: מספרי הקורסים ממוספים מ-1 עד 5 (ולא 0-4) ולכן יש להתאים את מספר הקורס לאינדקס הנכון במערך.

- **operator>>** עבור קלט של נתוני הסטודנט.
יש לקלוט לפי הסדר: מספר זהות, שם פרטי, שם משפחה ואחר כך 5 ערכים עבור 5 קורסים – התכונית תקלוט 0 עבור קורס שהסטודנט לא רשום עליו ו-1 עבור קורס שהסטוסנט כן רשום עליו
לדוגמה עבור הקלט הבא : 0 1 0 1 0
המשמעות היא שהסטודנט רשום לקורס מספר 2 וקורס מספר 4 (סופרים את הקורסים מ1 עד 5).

- **operator<<** עבור פלט של נתוני הסטודנט. ההדפסה תדפיס בשורה אחת את מספר הזהות, שם הפרטי, שם המשפחה ולאחר מכן רשימה של **מספרי הקורסים** שהסטודנט רשום אליהם.
לדוגמה אם סטודנט רשום רק לקורס מספר 2 וקורס מספר 5 אז יודפס: 5 2

ד. <u>כתוב תוכנית ראשית שבודקת את נכונות המחלקה שכתבת (אין צורך להגיש תוכנית זו).</u>

<u>שלב 2 -</u>

הגדר מחלקה בשם Registration המכילה קובץ בינארי, לניהול רישום סטודנטים לקורסים שונים.

יש לבנות את הקובץ כך שיכיל לכל היותר 100 סטודנטים. כל הרשומות זהות בגודלן ((sizeof(Student). מיקומו של כל סטודנט בקובץ יקבע לפי ערך id (מספר מ-1 עד 100) - כלומר סטודנט מספר i ישמר בקובץ במקום i (סטודנט מספר 1 יישמר ראשון בקובץ, סטודנט מספר 2 יישמר ברשומה השנייה בקובץ וסטודנט מספר 100 יישמר ברשומה האחרונה בקובץ).

תזכורת – הכתובת של הרשומה הראשונה בתוך הקובץ היא כתובת 0 (ולא 1).

לשם כך בשלב ראשון יש לבנות קובץ שמכיל 100 הרשומות ריקות – סטודנט ריק הוא סטודנט שמספרו (id) 0. במהלך התוכנית יש לאפשר גישה ישירה לסטודנטים, לפי דרישת המערכת.

א. הגדר מחלקה בשם Registration, השדה היחיד במחלקה יהיה:

- fileObj – אובייקט מטיפוס fstream דרכו ניתן לקרוא וכתוב נתונים מ/אל הקובץ.

ב. הוסף למחלקה 3 המתודות עזר פרטיות:

- `bool createFile(string);` – מתודה המקבלת שם של קובץ ומייצרת קובץ בינארי בשם זה עם מקום לאחסון נתונים של 100 סטודנטים.

תחילה יש לפתוח את הקובץ **לכתיבה** דרך אובייקט fileObj, לכתוב לתוכה 100 פעמים 'סטודנט ריק', ולבסוף לסגור הקובץ

במידה ויצירת הקובץ הצליחה יש להחזיר true ואחרת יש להחזיר false

- `Student readStudent(int);` - מתודה המקבלת מספר id של סטודנט, קוראת את הרשומה המתאימה מהקובץ ומחזירה אובייקט סטודנט (cbv).

<u>הערה</u>: בסיום הקריאה יש לקרוא למתודה fileObj.clear()

- `void writeStudent(Student);` - מתודה המקבלת סטודנט, וכותבת אותו לתוך הקובץ במקום המתאים לפי מספר הזהות שלו.

ג. הוסף למחלקה בנאי והורס לפי הפירוט הבא:

- **assignment constructor** בנאי המקבל מחרוזת עבור שם הקובץ שאותו רוצים לפתוח. יש לפתוח את הקובץ גם לכתיבה וגם לקריאה.

```
fileObj.open(fileName, ios::in | ios::out);
```

<u>שימו לב:</u> בפעם הראשונה, כאשר הקובץ עדיין לא קיים, פתיחה זו תכשל. לכן יש לבדוק תחילה שהקובץ קיים ובמידה והקובץ אינו קיים לייצר אותו בעזרת המתודה createFile ורק לאחר מכן לפתוח את הקובץ לקריאה וכתיבה.
<u>איך ניתן לבדוק האם קובץ כבר קיים?</u> ניתן לפתוח את הקובץ לקלט בלבד, ובמידה והפתיחה נכשלה אפשר להסיק שהקובץ אינו קיים.

```
Registration::Registration(string fileName)
{
    ____________________________________________//פתיחת הקובץ לקריאה בלבד
    if (___________)                            //בודק האם הקובץ עדיין לא קיים
        createFile(fileName);
    else
        ____________________________            //סגירת הקובץ

    ____________________________________________//פתיחת הקובץ לקריאה ולכתיבה
    if (____________) {                         //בודק אם הפתיחה נכשלה
        cout<< " could not open file\n";
    }

}
```

- **destructor** הורס – מתודה זו תפקידה לסגור את הקובץ בסיום העבודה.

ד. הוסף למחלקה את המתודות הציבוריות הבאות, הגדר את כל המתודות כך שאינן מקבלות פרמטרים:

- addStudent() - הוספת סטודנט – הפונקציה תדפיס enter student's details: ותקלוט מהמשתמש נתונים של סטודנט שאותו רוצים להוסיף לקובץ (יש להשתמש ב<<operator של Student).

על הפונקציה לבדוק שמספר הסטודנט שהתקבל ייחודי כלומר עדיין לא קיים בקובץ ולהוסיף את הסטודנט לקובץ במקום המתאים (לפי מספר הסטודנט).

במידה והסטודנט כבר קיים יש להדפיס student already exists ולצאת מהפונקציה.

- deleteStudent()- מחיקת סטודנט - על הפונקציה לבקש מהמשתמש להכניס את מספר הזהות של הסטודנט אותו רוצים למחוק מהקובץ enter id of student to delete – מחיקת רשומה נעשית החלפת הסטודנט בסטודנט ריק.

במידה והמחיקה הצליחה יש להדפיס את ת.ז. של הסטודנטי ואז ההודעה deleted אחרת יש להדפיס student does not exist

- update() - עדכון רישום סטודנט לקורס. על הפונקציה לבקש מהמשתמש להכניס את מספר הזהות של הסטודנט אותו רוצים לעדכן enter id of student to update

במידה והסטודנט לא קיים בקובץ יש להדפיס student does not exist.

אחרת יש לשאול את המשתמש לאיזה מספר קורס הוא מעוניין להירשם:

enter a course num to register for ולעדכן את הרשומה המתאימה בהתאם.

במידה והסטודנט לא קיים או שהמספר קורס שגוי אז נעשה שום שינוי לקובץ.

- checkRegistered() – מתודה ~~בוליאנית~~ שבודקת האם סטודנט מסוים רשום לקורס מסוים. על הפונקציה לבקשר מהמשתמש להכניס את מספר הזהות של הסטודנט ומספר הקורס שרוצים לבדוק (מספר בין 1 ל 5)

enter student id and course number

במידה שהסטודט רשום לקורס יש להדפיס

student X is registered for course Y

אחרת יש להדפיס

student X is not registered for course Y

כך ש-X הוא מספר הסטודנט ו-Y הוא מספר הקורס

- printStudent() - הדפסת פרטי סטודנט. על הפונקציה לבקש מהמשתמש מספר זהות של סטודנט enter student id number

במידה והסטודנט לא קיים בקובץ יש להדפיס student does not exist.

אחרת יש להדפיס נתוני הסטודנט (יש להתשתמש ב <<operator של Student).

- printAll() - הדפסת רשימה של כל סטודנטים הקיימים בקובץ. על הפונקציה לעבור על כל הקובץ כולו ולהדפיס למסך את נתוני כל הסטודנטים הקיימים בקובץ.

<u>הערה</u>: לא לשכוח לקורא ל fileObj.clear() בסיום הפונקציה.

ה. מצורף במודל קובץ בשם main.cpp עם תוכנית ראשית שמראה את נכונות המחלקה שכתבת.