תכנות מתקדם – 150024

תרגיל בית מספר 5

פונקציות חברות, string, חריגות וסטטי

<u>שים/י לב:</u>

א. הקפד/י על קְרִיאוּת התכנית ועל עימוד (Indentation).
ב. הקפד/י לבצע בדיוק את הנדרש בכל שאלה.
ג. בכל אחת מהשאלות יש להגדיר פונקציות במידת הצורך עבור קריאות התכנית.
ד. יש להגיש את התרגיל על פי ההנחיות להגשת תרגילים (המופיע באתר הקורס) וביניהם:
השתמש/י בשמות משמעותיים עבור המשתנים.
יש לתעד את התכנית גם עבור פונקציות אותם הנך מגדיר/ה וכן על תנאים ולולאות וקטעי קוד מורכבים, ובנוסף, **דוגמת הרצה לכל תכנית בסוף הקובץ**!
הגשה פרטנית.

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

<u>שאלה מס' 1:</u>

בשאלה זו עליך לכתוב תכנית לניהול כספומט.
בצע/י את השלבים הבאים, עד לתכנית המלאה.

בכל מצבי השגיאה האפשריים יש לזרוק חריגה מתאימה ולהמשיך לפעולה הבאה.
רשימת ההודעות לחריגה מפורטת בהמשך לגבי כל אחד מחלקי התכנית.
הנחיה כללית: כתוב כל קטע קוד במלואו עבור מצב תקין, ורק לאחר מכן הוסף את זריקת החריגות במקומות המתאימים.
ככלל – זריקת החריגות מתבצעת ע"י הפונקציות, ותפיסתן ע"י התכנית הראשית.
שים לב שעלייך לשנות בהתאם (כלומר – להוסיף תפיסת חריגות וכו') גם את התכנית הראשית.

תזכורת : אם זורקים מחרוזת מסוג rvalue (ערך הניתן להשמה במשתנה אחר, אך שלא ניתן לבצע אליו השמה - לדוגמא: "Wrong time format") אז חובה לתפוס אותה במשתנה מסוג const char *

<u>חלק א</u>

הגדר/י מחלקה **Clock** עבור ייצוג שעה.

א. המחלקה תכלול את התכונות הבאות:

- **hour** - השעה (0 עד 23)
- **minute** - מספר הדקות (0 עד 59)
- **second** - מספר השניות (0 עד 59)

יש להגדיר את תכונות המחלקה בזמן הגדרתן עם ערכי ברירת מחדל 0

ב. הוסף/י למחלקה לפחות את הבנאים הבאים:

- **constructor (בנאי עם פרמטרים)** -
- מקבל ערכים ומציב בתכונות.
במידה שאפילו אחד מהערכים שגוים אז
  - **כל התכונות** של המחלקה יישארו עם ערכי ברירת מחדל.
  - תיזרק הודעה מתאימה (הבנאי יזרוק את החריגה, שתתפס בתוכנית הראשית). במקרה זה תודפס רק הודעה אחת אפילו אם יש יותר משגיאה אחת.

הבנאי יבדוק את השגיאות ויזרוק שגיאות בסדר הבא:

Invalid time - negative number of seconds.
Invalid time - more than 60 seconds.
Invalid time - negative number of minutes.
Invalid time - more than 60 minutes.
Invalid time - negative number of hours.
Invalid time - more than 24 hours.

- כלומר קודם יבדוק האם יש שגיאה בשניות ואם כן תודפס הודעה מתאימה על השגיאה בשניות
- ואם אין שגיאה בשניות אז יבדוק האם יש שגיאה בדקות ואם כן תודפס הודעה מתאימה על השגיאה בדקות
- ואם אין שגיאה בשניות ודקות אז יבדוק האם יש שגיאות בשעות ואם כן תודפס הודעה מתאימה על השגיאה בשעות.

- ***שאלה למחשבה – למה אין פה copy constructor (בנאי העתקה)?***

ג. הוסף/י למחלקה את המתודות הבאות:

- הצבה ואחזור (get/set) לכל תכונה. עבור כל set, במידה שהתקבל ערך שגוי יש להשאיר את התכונה בלי לשנות אותה – אבל כן יש צורך לזרוק שגיאה כך שהתוכנית הראשית תדפיס הודעה מתאימה.
- פונקציה בוליאנית `equals`, המקבלת שעה ובודקת אם היא זהה לשעה המעודכנת באובייקט.
- פונקציה בוליאנית `before`, המקבלת שעה ובודקת אם השעה המעודכנת באובייקט היא לפני השעה שהתקבלה.
- פונקציה בוליאנית `after`, המקבלת שעה ובודקת אם השעה המעודכנת באובייקט היא אחרי השעה שהתקבלה.
- **operator+=** המקבל כפרמטר מספר שניות ומקדם את השעות בהתאם.
- **operator<<** להדפסת השעה בפורמט hh:mm:ss
(שים/י לב שגם כאשר הערכים קטנים מ-10 יש להדפיס בפורמט של 2 ספרות).
- **operator>>** עבור קלט בפורמט המתאים. במידה והקלט אינו תקין תשלח הודעת חריגה ובאובייקט יוצב הערך 00:00:00.

בכל מקרה של שגיאה תיזרק חריגה מתאימה לפי הניסוח בפירוט הבא:

Invalid time - negative number of seconds.
Invalid time - more than 60 seconds.
Invalid time - negative number of minutes.
Invalid time - more than 60 minutes.
Invalid time - negative number of hours.
Invalid time - more than 24 hours.

במידה שיש שגיאה ב שעון והשגיאה לא נופלת באחת מ-6 הקטגוריות הקודמות יש להדפיס :
Wrong time format.

חלק ב

הגדר/י מחלקה **Account** עבור ייצוג חשבון בנק.

א. המחלקה תכלול את התכונות הבאות :

- **accountNumber** - מספר החשבון.
- **code** - קוד סודי בן 4 ספרות (הספרה השמאלית ביותר אינה 0).
- **balance** – היתרה בחשבון.
- **email** – כתובת מייל של בעל החשבון - מטיפוס string – כתובת תקינה מכילה @ ולפניו רצף של תווים ללא רווחים ולאחריו רצף תווים ללא רווחים עם אחת מהסיומות הבאות : .com או .co.il
(יש להשתמש במחלקת string המוגדרת ב - STL)
https://www.cplusplus.com/reference/string/string/

ב. הוסף/י למחלקה לפחות את הבנאים הבאים :

- **empty constructor (בנאי ריק)** – המציב 0 בכל התכונות ומחרוזת ריקה בתכונת המייל.
- **constructor (בנאי עם פרמטרים)** - מקבל ערכים ומציב בתכונות, במידה שאפילו אחד מהערכים שגוים אז
  - **כל התכונות** של המחלקה יישארו עם ערכי ברירת מחדל.
  - תיזרק הודעה מתאימה (הבנאי יזרוק את החריגה, שתתפס בתוכנית הראשית). במקרה זה תודפס רק הודעה אחת אפילו אם יש יותר משגיאה אחת.
    - אם יש שגיאה בקוד סודי אז יודפס : ERROR: code must be of 4 digits!
    - ואם אין שגיאה בקוד סודי אבל יש שגיאה במייל שחסר סימן @ אז יודפס : ERROR: email must contain @!
    - ואם אין שגיאה בקוד סודי וגם אין שגיאה עם הסימן @ אבל יש שגיאה במייל כי סיומת שגויה אז יודפס :

ERROR: email must end at .com or .co.il!

ג. הוסף/י למחלקה את המתודות הבאות :

- פונקציות אחזור (get) לכל תכונות המחלקה. אין דרך לשנות את תכונות המחלקה אחרי שהתקבלו ערכים על ידי העברת נתונים לבנאים, (אלא ע"י אופרטור הקלט) לכן אין צורך בפונקציות של set.
- **operator<<** - עבור קלט של נתוני החשבון ההתחלתיים - מספר חשבון קוד סודי וכתובת מייל. הלקוח אינו מזין יתרה. היתרה הראשונית הינה 0. במידה שאפילו אחד מהערכים שגוים אז הערכים של **כל התכונות** של המחלקה לא ישתנו ותיזרק הודעה מתאימה (תיזרק חריגה בפונקציה ותיתפס בתוכנית הראשית). במקרה זה תודפס רק הודעה אחת אפילו אם יש יותר משגיאה אחת.
  - אם יש שגיאה בקוד סודי אז יודפס : ERROR: code must be of 4 digits!

- אם אין שגיאה בקוד הסודי אבל יש שגיאה במייל שחסר סימן @ אז יודפס:
  ERROR: email must contain @!
- אם אין שגיאה בקוד הסודי ואין שגיאה בסימן @ אבל יש שגיאה במייל שהסיומת שגויה אז יודפס:
  ERROR: email must end at .com or .co.il!

- **withdraw(int nis)** – מתודה למשיכת מזומנים בסכום של nis = ש״ח (₪). בפעולה אחת ניתן למשוך מזומנים עד לסכום של 2500 ₪, עם מסגרת אשראי של 6000 ₪ (כלומר תאפשר הגעה למינוס של עד 6000 ₪)
  קודם יש לבדוק האם יש מינוס של יותר מ 6000 ₪ ובמידה שכן זורק חריגה, אחר כך יש לבדוק האם הסכום מעל 2500 ₪ במידה שכן זורק חריגה. אם הכל תקין אז ממשיך עם הפעולה (במידה ויש חריגה - לא מתבצעת משיכה, כמובן).
- **deposit(int)** – מתודה המאפשרת להפקיד צ׳קים בחשבון בסכום שאינו עולה על 10000 ₪. במידה שסכום הצ׳ק מעל 10000 ₪ אז נזרקת חריגה וההפקדה לא מתבצעת.

בנוסף, המחלקה תכלול תכונות ומתודות סטטיים:

- **sumWithdraw** – תכונה: מונה של סכום כל המשיכות מהבנק (מכל החשבונות).
- **sumDeposit** – תכונה: מונה של סכום כל ההפקדות בהמחאה (מכל החשבונות).
- **getSumWithdraw()** – מתודה המחזירה את הסכום כל המשיכות מכל החשבונות יחד.
- **getSumDeposit()** - מתודה המחזירה את הסכום כל הצ׳קים שהופקדו עד עכשיו.

בכל פעם שמתעוררת בעיה יש לזרוק הודעת שגיאה מתאימה.
להלן נוסח ההודעות:

ERROR: wrong code! //קוד שגוי
ERROR: wrong email! //כתובת מייל שגויה
ERROR: cannot deposit more than 10000 NIS!//אסור להפקיד יותר מ 1000
ERROR: cannot withdraw more than 2500 NIS!//אסור למשוך יותר מ2500
ERROR: cannot have less than - 6000 NIS!//אסור מינוס יותר מ6000

חלק ג

נתונה התכנית הראשית הבאה למימוש כספומט. התכנית משתמשת במחלקות שהגדרת בשאלות הקודמות, בתכנית זו מניחים שהבנק כולל 10 חשבונות בלבד.

בשלב ראשון המשתמש נדרש להזין את נתוני כל החשבונות:

- מספר חשבון (המספר חייב להיות ייחודי, כלומר – לא יהיו שני חשבונות עם אותו מספר חשבון),
- קוד סודי (בן 4 ספרות, הספרה השמאלית אינה 0).
- כתובת email (רצף תווים ללא רווחים, מכיל @ אחד בלבד עם אחת מהסיומות: .com או .co.il)

המשתמש אינו מזין את היתרה. נניח שהיתרה ההתחלתית היא תמיד 0.

החריגות האפשריות בשלב זה:

ERROR: code must be of 4 digits!//קוד חייב להיות בן 4 ספרות
ERROR: account number must be unique!//מספר חשבון חייב להיות ייחודי
ERROR: email must contain @!//מייל חייב להכיל @
ERROR: email must end at .com or .co.il!//מייל חייב להכיל סיומת תקינה

לאחר מכן התכנית מאפשרת לבחור בלולאה בסדרה של פעולות עד לבחירת 0 – stop.
בכל שלב התכנית אמורה לבצע פעולה אחת ולהדפיס הודעה מתאימה כולל את שעת ביצוע הפעולה.
במידה והתקבלה חריגה התוכנית אמורה להדפיס את השעה (לפני ביצוע הפעולה) ואת תוכן החריגה שהתקבלה ותמשיך לבקש את הבחירה הבאה.

נניח כי שעת פתיחת הכספומט הינה שמונה בבוקר 08:00:00
בירור יתרה אורך 20 שניות,
משיכת מזומנים אורך 50 שניות,
הפקדה אורכת 30 שניות,
הצגת סך ההפקדות או סך כל המשיכות אורכת דקה.
נניח כי כל הפעולות מתבצעות ברצף, ללא הפסקה.

שים/י לב: <u>החריגות נזרקות מהמתודות</u> המתאימות (ולא נכתבות בתכנית הראשית אלא נתפסות בתכנית הראשית). את החריגות יש לבצע <u>בתחילת המתודה</u> המתאימה.
את החריגות יש לתפוס בתכנית הראשית במקומות המתאימים. כלומר, <u>יש להוסיף בתכנית הראשית הנתונה תפיסות לחריגות</u> האפשריות.

(לצורך הדגמה – נכתב בתוכנית בלוק try-catch אחד. הוסף/י את השאר במקומות המתאימים)

```
#include <iostream>
#include "Clock.h"
#include "Account.h"

using namespace std;

enum ACTION {
	STOP,
	BALANCE,
	DEPOSIT,
	WITHDRAW,
	SUM_DEPOSIT,
	SUM_WITHDRAW
};
ACTION menu() {
	cout << "enter 1 to get account balance" << endl;
	cout << "enter 2 to deposit money" << endl;
	cout << "enter 3 to withdraw money" << endl;
	cout << "enter 4 to see the sum of all deposits" << endl;
	cout << "enter 5 to see the sum of all withdrawals" << endl;
	cout << "enter 0 to stop" << endl;
	int x;
	cin >> x;
	return (ACTION)x;
}
int findAccount(Account* bank, int size) {
	int number, code;
	cout << "please enter account number:\n";
	cin >> number;
	int i = 0;
	while (i < size && bank[i].getAccountNumber() != number)
		i++;
	if (i >= size)
		throw "ERROR: no such account number\n";
	cout << "please enter the code:\n";
```

```
        cin >> code;
        if (bank[i].getCode() == code)
                return i;
        throw "ERROR: wrong code!\n";
}
void printTransaction(Account a, ACTION ac, Clock& c) {
        cout << c << "\t";
        switch (ac) {
        case BALANCE: cout << "account #: " << a.getAccountNumber() << "\t";
                cout << "balance: " << a.getBalance() << endl;
                break;
        case DEPOSIT:
        case WITHDRAW: cout << "account #: " << a.getAccountNumber() << "\t";
                cout << "new balance: " << a.getBalance() << endl;
                break;
        case SUM_DEPOSIT:
                cout << "sum of all deposits: " << Account::getSumDeposit() << endl;
                break;
        case SUM_WITHDRAW:
                cout << "sum of all withdrawals: " << Account::getSumWithdraw() <<
endl;
                break;
        }
}
void getBalance(Account* bank, int size, Clock& c) {
        int i = findAccount(bank, size);
        c += 20;
        printTransaction(bank[i], BALANCE, c);
}
void cashDeposit(Account* bank, int size, Clock& c) {
        int i = findAccount(bank, size);
        float amount;
        cout << "enter the amount of the deposit:\n ";
        cin >> amount;
        bank[i].deposit(amount);
        c += 30;
        printTransaction(bank[i], DEPOSIT, c);
}
void cashWithdraw(Account* bank, int size, Clock& c) {
        int i = findAccount(bank, size);
        float amount;
        cout << "enter the amount of money to withdraw:\n ";
        cin >> amount;
        bank[i].withdraw(amount);
        c += 50;
        printTransaction(bank[i], WITHDRAW, c);
}
void checkAccount(Account bank[], int i) {
        for (int j = 0; j < i; j++)
                if (bank[i].getAccountNumber() == bank[j].getAccountNumber())
                        throw "ERROR: account number must be unique!\n";
}
int main() {
                const int SIZE = 10;
                Clock c(8, 0, 0);
                Account bank[SIZE];
                cout << "enter account number, code and email for " << SIZE << "
accounts:\n";
```

```
        for (int i = 0; i < SIZE; i++) {
            try {
                cin >> bank[i];
                checkAccount(bank, i);
            }
            catch (const char* msg) {
                cout << c << '\t' << msg;
                i--;
            }
        }
        ACTION ac = menu();
        while (ac) {
                switch (ac) {
                case BALANCE: getBalance(bank, SIZE, c);
                    break;
                case WITHDRAW:cashWithdraw(bank, SIZE, c);
                    break;
                case DEPOSIT:cashDeposit(bank, SIZE, c);
                    break;
                case SUM_DEPOSIT:c += 60;
                    printTransaction(bank[0], SUM_DEPOSIT, c);
                    break;
                case SUM_WITHDRAW:c += 60;
                    printTransaction(bank[0], SUM_WITHDRAW, c);
                }

            ac = menu();
        }

    return 0;
}
```

דוגמאה להרצת התכנית (תוכנית עם רק 3 חשבונות ובלי שגיאות):

```
enter account number, code and email for 3 accounts:
123  4444 me@gmail.com
 5555 234 you@walla.co.il
 6666 345 us@g.com
enter 1 to get account balance
enter 2 to deposit money
enter 3 to withdraw money
enter 4 to see the sum of all deposits
enter 5 to see the sum of all withdrawals
enter 0 to stop
2
please enter account number:
234
please enter the code:
5555
enter the amount of the deposit:
5000
08:00:30        account #: 234  new balance: 5000
enter 1 to get account balance
enter 2 to deposit money
enter 3 to withdraw money
enter 4 to see the sum of all deposits
enter 5 to see the sum of all withdrawals
enter 0 to stop
3
please enter account number:
234
please enter the code:
5555
enter the amount of money to withdraw:
1000
08:01:20 account #: 234  new balance: 4000
```

```
enter 1 to get account balance
enter 2 to deposit money
enter 3 to withdraw money
enter 4 to see the sum of all deposits
enter 5 to see the sum of all withdrawals
enter 0 to stop
1
please enter account number:
234
please enter the code:
5555
08:01:40        account #: 234  balance: 4000
enter 1 to get account balance
enter 2 to deposit money
enter 3 to withdraw money
enter 4 to see the sum of all deposits
enter 5 to see the sum of all withdrawals
enter 0 to stop
2
please enter account number:
345
please enter the code:
6666
enter the amount of the deposit:
2000
08:02:10        account #: 345  new balance: 2000
enter 1 to get account balance
enter 2 to deposit money
enter 3 to withdraw money
enter 4 to see the sum of all deposits
enter 5 to see the sum of all withdrawals
enter 0 to stop
```

```
3
please enter account number:
345
please enter the code:
6666
enter the amount of money to withdraw:
500
08:03:00        account #: 345  new balance: 1500

enter 1 to get account balance
enter 2 to deposit money
enter 3 to withdraw money
enter 4 to see the sum of all deposits
enter 5 to see the sum of all withdrawals
enter 0 to stop
4
08:04:00 sum of all deposits: 7000
enter 1 to get account balance
enter 2 to deposit money
enter 3 to withdraw money
enter 4 to see the sum of all deposits
enter 5 to see the sum of all withdrawals
enter 0 to stop
5
08:05:00 sum of all withdrawals: 1500
enter 1 to get account balance
enter 2 to deposit money
enter 3 to withdraw money
enter 4 to see the sum of all deposits
enter 5 to see the sum of all withdrawals
enter 0 to stop
0
```

בהצלחה!!