סדנא ב- C++ – 150018

תרגיל בית מספר 4

פונקציות חברות, 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 יחידנית - אין להגיש בזוגות.

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.

א. הגדר/י מחלקה בשם **Clock** לייצוג שעה.
התכונות במחלקה יהיו:
- **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

הוסף/י למחלקה את המתודות הבאות:
א. Constructors:
1. default -הקובע את השעה 00:00:00.
2. קונסטרקטור עם פרמטרים - מקבל ערכים ומציב בתכונות, במידה והערכים שגויים השעה תיקבע ל- 00:00:00.
3. copy constructor
ב. set/get עבור כל תכונה במחלקה
ג. אופרטור =+ המקבל כפרמטר מספר שניות ומקדם את השעות בהתאם.
ד. אופרטור פלט >> להדפסת השעה בפורמט hh:mm:ss (שים/י לב שגם כאשר הערכים קטנים מ-10 יש להדפיס בפורמט של 2 ספרות).

ה. אופרטור קלט << עבור קלט בפורמט המתאים. במידה והקלט אינו תקין תשלח הודעת חריגה ובאובייקט יוצב הערך 00:00:00.

בכל מקרה של חריגה תשלח הודעה מתאימה לפי הניסוח בפירוט הבא:

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

ב. הגדר/י מחלקה בשם **Account** לייצוג חשבון בנק.
התכונות במחלקה יהיו:

- **accountNumber -** מספר החשבון.
- **code -** קוד סודי בן 4 ספרות (הספרה השמאלית ביותר אינה 0).
- **balance** – היתרה בחשבון.
- **mail** – 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/

הוסף/י למחלקה את המתודות הבאות:

א. constructors:
   1. default -המציב 0 בכל התכונות ומחרוזת ריקה בתכונת המייל.
   2. קונסטרקטור עם פרמטרים - מקבל ערכים ומציב אותם בתכונות.

ב. set/get לפי הצורך (אין לאפשר לשנות את הקוד הסודי ואת מספר החשבון)

ג. אופרטור << עבור קלט של נתוני החשבון ההתחלתיים - מספר חשבון קוד סודי וכתובת מייל. הלקוח אינו מזין יתרה. היתרה הראשונית הינה 0.

ד. **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 חריגה של עד 6000 ₪)

ה. **deposit(int)** – מתודה המאפשרת להפקיד צ'קים בחשבון בסכום שאינו עולה על 10000 ₪.

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).

a. **getSumWithdraw()** – מתודה לברור סכום כל המשיכות מכל החשבונות יחד.
b. **getSumDeposit()** - מתודה לברור סכום 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

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.

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

using namepace std;

enum action
{
	stop,
	balance,
	deposit,
	withdraw,
	sumDeposit,
	sumWithdraw
};
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 withdraws" << endl;
	cout << "enter 5 to see the sum of all Deposits" << 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: ";
	cin >> number;
	int i = 0;
	while (i < size && bank[i].getAccountNumber() != number)
		i++;
	cout << "please enter the code: ";
	cin >> code;
	if (bank[i].getCode() == code)
		return i;
}
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 sumDeposit:
		cout << "sum of all deposits: " << Account::getSumDeposit() << endl;
		break;
	case sumWithdraw:
```

```
		cout << "sum of all withdraws: " << 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 check: ";
	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: ";
	cin >> amount;
	bank[i].withdraw(amount);
	c += 50;
	printTransaction(bank[i], withdraw, c);
}
int main()
{
	Clock c(8);
	Account bank[10];
	cout << "enter account number, code and email for 10 accounts:\n";
	for (int i = 0; i < 10; i++)
	{
		try {
			cin >> bank[i];
			for (int j = 0; j < i; j++)
				if (bank[i].getAccountNumber() ==
bank[j].getAccountNumber())
					throw "ERROR: account number must be unique!\n";
		}
		catch (const char* msg)
		{
			cout << c << '\t' << msg;
			i--;
		}
	}
	action ac = menu();
	while (ac)
	{
		switch (ac)
		{
		case balance: getBalance(bank, 10, c);
			break;
```

```
            case withdraw:cashWithdraw(bank, 10, c);
                  break;
            case deposit:cashDeposit(bank, 10, c);
                  break;
            case sumDeposit:c += 60;
                  printTransaction(bank[0], sumDeposit, c);
                  break;
            case sumWithdraw:c += 60;
                  printTransaction(bank[0], sumWithdraw, c);

            }
            ac = menu();
      }
      return 0;
}
```

בהצלחה!!