סדנא ב- C++ – 150018

תרגיל בית מספר 8

תור ומחסנית

שים/י לב:

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 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.

שים/י לב: בתרגיל זה אסור להשתמש ב-STL!

שאלה מס' 1:

בשאלה זו נגדיר תכנית אשר תקלוט מהמשתמש ביטוי מתמטי (infix) **כמחרוזת**, והתכנית תחשב את ערך הביטוי תוך שמירה על סדר פעולות חשבון. נעשה זאת באמצעות שימוש במחסנית.
חובה לפתור את השאלה הזאת על ידי שימוש ב- template

תזכורת : במחסנית רק עובדים על ה- top של מחסנית
זאת אומרת שאם אומרים ל"דחוף" אז דוחפים לראש המחסנית (push)
אם מסתכלים על מה שיש במחסנית אז הכוונה שבודקים את מה שיש בראש המחסנית (top)
ואם שולפים מהמחסנית אז רק שולפים את מה שיש בראש המחסנית (pop)

א. הוסף לתוכנית את המחלקה Vector התבניתית (template) שהוצגה בכיתה (ובדוגמאות הקוד).(כך - שכשנגדיר מופע של המחלקה בהמשך התרגיל, יהיה ניתן לבנות את הוקטור כך שיכיל תוים (char) ואפשר גם לבנות את הוקטור כך שיכיל מספרים שלמים (int))
- הוסף למחלקה Vector<T> מתודה בשם isEmpty שתחזיר true במידה שהוקטור ריק. יש לכתוב את המתודה כ- const
כלומר להשתמש בחתימה הבאה:

```
template <class T> bool Vector<T>::isEmpty() const
```

ב. הוסף לתוכנית מחלקה תבניתית (template) בשם StackVector<T>. המחלקה תהיה דומה למחלקה StackVector שהוצגה בכיתה (ובדוגמאות הקוד) אבל הפעם היא גם תהיה template.
- למחלקה ישנה תכונה אחת בלבד - data - שהיא מופע של המחלקה Vector<T>

- יש להתאים לפי צורך את כל המתודות של המחלקה כך שתהיינה מתאימות לתכונה מסוג Vector<T> (במקום רק Vector)
- כשתכתבו את הפונקציה של infixToPostfix – תשתמשו במחסנית של StackVector<char> (פרטים בהמשך)
- כשתכתבו את הפונקציה של calcPostfix – תשתמשו במחסנית של StackVector<int> (פרטים בהמשך)

ג. הגדר/י פונקציה גלובאלית בשם infixToPostfix המקבלת כמחרוזת ביטוי בייצוג infix ומחזירה מחרוזת חדשה עם ביטוי זהה בייצוג postfix (האלגוריתם מובא בהמשך)
לדוגמא: עבור הקלט

$$"(\mathbf{5}+\mathbf{3})*((\mathbf{20}/\mathbf{10})+(\mathbf{8}-\mathbf{6}))"$$

הפונקציה תחזיר:

$$"\mathbf{5\ 3\ +\ 20\ 10\ /\ 8\ 6\ -\ +\ *}"$$

שים/י לב:

- הגרשיים אינם חלק מהביטוי
- הביטוי יכול להכיל מספרים בעלי יותר מספרה אחת
- הצב/י ' ' (רווח בודד) בין כל שני רכיבים בביטוי postfix המתקבל

שים/י לב: בפונקציה infixToPostfix המחסנית מיועדת להחזקה זמנית של של תווים (char). לכן, השתמשו ב StackVector<char> כלומר תפעילו את StackVector- כך שהתכונה data תהיה מסוג Vector<char> (תכיל תווים).

<u>המרת ביטוי infix (תוכי) לביטוי postfix (סופי):</u>

אלגוריתם הקולט ביטוי בייצוג infix - תוכי, תו אחר תו וממיר אותי לביטוי בייצוג postfix – סופי
הנחה: הביטוי הנקלט מכיל סוגריים עגולים בלבד (כולל סוגריים מקוננים)

1. צור/י מחרוזת ריקה $str \leftarrow$
2. צור/י מחסנית-תווים ריקה $stk \leftarrow$
3. קרא/י תו ראשון מהקלט $ch \leftarrow$
4. כל עוד לא נגמר הקלט, בצע/י:

4.1 אם (ch סוגר פותח - שמאלי)
דחוף/י את ch למחסנית

4.2 אם (ch סוגר סוגר - ימני)
שלוף/י את תוכן המחסנית עד (לא כולל) הסוגר השמאלי ושרשר/י למחרוזת str.
בסיום – שלוף/י את הסוגר השמאלי.

4.3 אם (ch אופרטור –פעולה חשבונית)
שלוף/י מהמחסנית את כל הפעולות בדרגת קדימויות גבוהה יותר ושרשר/י אותן למחרוזת str. (ממשיכים לשלוף כל עוד יש פעולה בדרגת קדימויות גבוהה יותר או

עד שמגיעים לסוגר פותח – שמאלי)
בסיום – דחוף/י את ch למחסנית.
4.4 אם (ch ספרה / מספר)
שרשר/י אותו למחרוזת str

4.5 קרא/י את התו הבא מהקלט ← ch
5. כל עוד המחסנית אינה ריקה
שלוף/י ושרשר/י את תוכן המחסנית למחרוזת str

תזכורת: חובה לשים רווח בין כל שני רכיבים – לכן, כאשר משרשרים ערכים למחרוזת *str* חובה להוסיף רווחים

- בסעיפים 4.2, 4.3 ו-5 אחרי הוספת כל ערך למחרוזת - יש להוסיף רווח
- בסעיף 4.4 אחרי הוספת מספר (**ולא ספרה**) למחרוזת - יש להוסיף רווח
- אחרי הוספת מספר האחרון למחרוזת יש להוסיף רווח

ד. הגדר/י פונקציה גלובאלית בשם calcPostfix המקבלת כפרמטר **מחרוזת** עם ביטוי חשבוני בייצוג postfix (סופי) (הביטוי יכול להכיל פעולות חיבור, חיסור, כפל וחילוק) על הפונקציה לחשב ולהחזיר את תוצאת הביטוי. (האלגוריתם מובא בהמשך)

לדוגמא עבור הקלט

$$\mathbf{5\ 3\ +\ 20\ 10\ /\ 8\ 6\ -\ +\ *}$$

הפונקציה תחזיר: 32

שים/י לב: בפונקציה calcPostfix המחסנית מיועדת להחזקה זמנית של מספרים שלמים (int). ולכן, השתמשו ב StackVector<int> כלומר תפעילו את StackVector- כך שהתכונה data תהיה מסוג Vector<int> (תכיל מספרים שלמים).

<u>חישוב ביטוי postfix (סופי) באמצעות מחסנית:</u>

אלגוריתם לחישוב ביטוי postfix – סופי
1. התחל/י עם מחסנית ריקה

2. עבור/י על כל ביטוי משמאל לימין:
3. אם האיבר הבא הוא אופרנד – הכנס/י אותו למחסנית
4. אם הוא פעולה – הפעל/י את הפעולה על שני האיברים הנמצאים בראש המחסנית והכנס/י את התוצאה למחסנית

ה. להלן תכנית ראשית הבודקת את נכונות התכנית:

```
int main() {

	string exp;
	cout << "enter an infix expression as a string" << endl;
	cin >> exp;
	string postfix = infixToPostfix(exp);
	cout << "in postfix form: " << postfix << endl;
	cout << "calculated value: " << calcPostfix(postfix) << endl;
	return 0;
}
```

<u>שאלה מס' 2:</u>

בהרצאה הוגדרה המחלקה האבסטרקטית Queue באופן הבא:

```
class Queue
{
public:
```

```
    virtual void clear() = 0;
    virtual void enqueue(int value) = 0;
    virtual int dequeue() = 0;
    virtual int front ()  = 0;
    virtual bool isEmpty() const = 0;
};
```

עבור מחלקה זו הוצעו בכיתה שתי הצעות למימושים שונים לתור, אחת ע"י מערך ($Vector$) והשניה ע"י רשימה ($List$). בשאלה זו נדרוש מימוש נוסף (שלישי) והפעם בעזרת מחסנית (ומחסנית עזר נוספת):

```
class QueueStack : public Queue
{
protected:
      Stack* data;
public:
       QueueStack();
        void clear() override;
       int dequeue() override ;
       void enqueue(int value) override;
       int front() override;
       bool isEmpty() const override;
};
```

עליך לממש מחלקה זו.

רמז:
המחלקות שיכולות לעזור לך לכתוב את הפתרון:

1. מחלקת Queue המוגדרת כפי שרשום למעלה (שימו לב: תצטרך/י להגיש רק את הקובץ Queue.h עבור מחלקה זו כי אין מימוש לאף אחת מהמתודות שלה)
2. מחלקת QueueStack (גם קובץ h וגם קובץ cpp)
   עליך לממש מחלקה זו כך שהמחסנית תתנהג כתור (FIFO – ראשון הנכנס ראשון היוצא) למרות שבפועל התכונה היחידה הנמצאת בה ,היא מופע של Stack (מחסנית).
   יש להיעזר במחסנית עזר, אליה ניתן "לרוקן" את המחסנית המקורית, במקרה של צורך להגיע לאיבר שנמצא בתחתית המחסנית. כמובן יש להחזיר את האיברים למחסנית המקורית אחרי הוצאת האיבר מהתחתית.
3. מחלקת Stack (גם קובץ h וגם קובץ cpp)
   ניתן לכתוב את המחלקה הזו לבד או להשתמש במחלקת Stack שלמדת בהרצאה.
   אם בחרת להשתמש במחלקה שהודגמה בהרצאה – אז עליך לבחור האם המימוש יהיה על ידי רשימה List או על ידי Vector
4. אם בסעיף 3 – החלטת להשתמש במחלקה Stack של ההרצאה
   a. אם בחרת במימוש על ידי רשימה אז יש להוסיף את מחלקת List (גם קובץ h וגם קובץ cpp)
   b. אם בחרת במימוש על ידי וקטור אז יש להוסיף את מחלקת Vector (גם קובץ h וגם קובץ cpp)

להלן תכנית ראשית הבודקת את נכונות המחלקה:

```
#include <iostream>
#include "QueueStack.h"
using namespace std;
int main() {
	Queue* Q;
	int num;
	Q = new QueueStack();
	cout << "enter 5 numbers: "<< endl;
	try {
		for (int i = 0; i < 5; i++) {
			cin >> num;
			Q->enqueue(num);
		}
	}
	catch (const char* msg)
	{
		cout << msg;
	}
	cout << "first in queue is: " << Q->front() << endl;
	cout << "removing first 2 elements:" << endl;
	cout << Q->dequeue() << ' ';
	cout << Q->dequeue() << endl;
	cout << "first in queue is: " << Q->front() << endl;
	Q->enqueue(8);
	Q->enqueue(9);
	cout << "emptying out the queue: " << endl;
	while (!Q->isEmpty())
		cout << Q->dequeue() << " ";
	cout << endl;
	return 0;
}
```

דוגמא להרצת התכנית:

```
enter 5 numbers:
1 2 3 4 5
first in queue is: 1
removing first 2 elements:
1 2
first in queue is: 3
3 4 5 8 9
```

בהצלחה רבה!