**תרגיל 8**

**תור ומחסנית**

1. בתרגיל זה נרצה לכתוב תוכנית אשר תקלוט מהמשתמש ביטוי מתמטי (infix) **כמחרוזת**, ותחשב את ערך הביטוי תוך שמירה על סדר פעולות חשבון.
2. כתוב פונקציה בשם infixToPostfix שמקבלת במחרוזת ביטוי בייצוג infix ומחזירה מחרוזת חדשה עם אותו הביטוי אבל הפעם בייצוג postfix. (האלגוריתם נתון בהמשך).

לדוגמא עבור הקלט

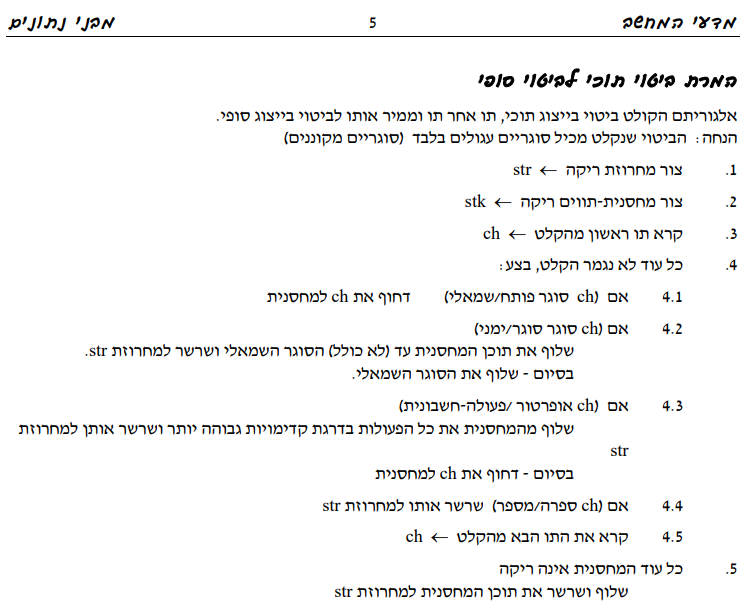
**"(5+3)\*((20/10)+(8-6))"**

הפונקציה תחזיר:

**"5 3 + 20 10 / 8 6 - + \*"**

שימו לב-

1. הגרשיים אינם חלק מהביטוי
2. הביטוי יכול להכיל מספרים בעלי יותר מספרה אחת .
3. הציבו ' ' (רווח בודד) בין כל שני רכיבים בביטוי postfix שמתקבל.

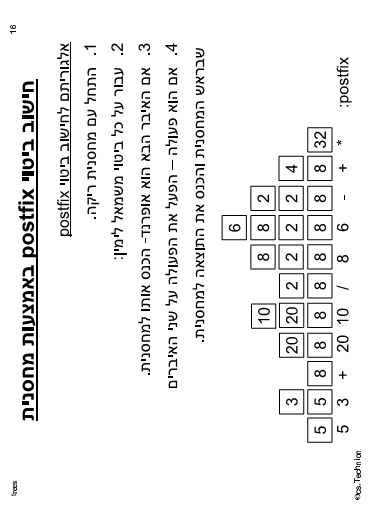


1. כתוב פונקציה בשם CalcPostfix שמקבלת כפרמטר **מחרוזת** עם ביטוי חשבוני בייצוג סופי (postfix) (הביטוי יכול להכין פעולות חיבור, חיסור, כפל וחילוק) על הפונקציה לחשב ולהחזיר את תוצאת הביטוי. (האלגוריתם נתון הבמשך).

לדוגמא עבור הקלט

1. **3 + 20 10 / 8 6 - + \***

הפונקציה תחזיר: 32



1. השתמשו בתוכנית הראשית הבאה כדי לבדוק את נכונות התוכנית שכתבתם.

int main()

{

string exp;

cout << "enter an infix expression as a string" << endl;

cin >> exp;

string postfix = infixToPostfix(exp);

cout << postfix<<endl;

cout << calcPostfix(postfix)<<endl;

return 0;

}

2. בשיעור הגדרנו את המחלקה האבסטרקטית Queue כך:

template <class T>

class Queue

{

public:

virtual void clear() = 0;

virtual T enqueue() = 0;

virtual void dequeue(int value) = 0;

virtual T front () = 0;

virtual bool isEmpty() const = 0;

};

ולאחר מכן נתנו 2 מימושים שונים לתור, אחת ע"י מערך והשנייה ע"י רשימה. בתרגיל זה נרצה לתת לתור ממוש שלישי והפעם בעזרת מחסנית.

|  |
| --- |
| template <class T>  class QueueStack : public Queue<T>  {  protected:  stack<T> data;  public:  QueueStack();  void clear() override;  T dequeue() override ;  void enqueue(T value) override;  T front() override;  bool isEmpty() const override;  }; |

הגישו את התרגיל בצירוף התוכנית הראשית הבאה:

#include <iostream>

#include "Header.h"

using namespace std;

int main() {

Queue<int>\* Q;

Q= new StackQueue<int>();

try {

for (int i = 0; i < 10; i++)

Q->enqueue(i);

}

catch (const char\* msg)

{

cout << msg;

}

cout << "first on Q is: " << Q->front() << endl;

cout << "take out 2 elemets:" << endl;

cout << Q->dequeue() << ' ' << Q->dequeue() << endl;

cout << "first on Q is: " << Q->front() << endl;

Q->enqueue(8);

Q->enqueue(9);

while (!Q->isEmpty())

cout << Q->dequeue() << " ";

return 0;

}