מבני נתונים
תרגיל 4 - פתרונות

תורים ומחסניות

1. כתבו את המימוש של תור בעזרת מערך, כפי שעשינו בכיתה, אך הוסיפו את בדיקת מקרי הקצה. ממשו את שלוש השיטות:

```
void enqueue(int i);
int dequeue();
boolean isEmpty();
```

הרגישו חופשיים לשנות את פרטי המימוש כרצונכם.

פתרון: יש פה הרבה פתרונות אפשריים. מעבר לבדיקה הרגילה, הבעיה היא שקשה במימוש כמו שהוא להבדיל בין המקרה בו התור ריק והמקרה בו הוא מלא. אחד הפתרונות הפשוטים הוא להוסיף משתנה המחזיק את מס' האברים בתור בכל שלב. זה הפתרון שמתואר פה.

```
int[] a = new int[SIZE];
int head = 0;
int tail = 0; // is actually after the tail.
int size = 0;

void enqueue(int i)
{
  if (size == SIZE)
    ERROR overflow
  size++;
  a[tail] = i;
  tail++;
  if (tail >= SIZE)
    tail = 0;
}
```

```
int dequeue()
{
  if (size == 0)
    ERROR underflow
  size++;
  int x = a[head];
  head++;
  if (head >= size)
    head = 1;
  return x;
}
boolean isEmpty()
{
 return (size == 0);
}
```

2. הסבירו כיצד ניתן לממש שתי מחסניות בתוך מערך אחד כך שכל עוד סכום שתיהן קטן מגודל המערך, אף אחת מהן לא מגיע ל overflow. כיתבו קוד ל־4 השיטות

```
void push1(int i);
void push2(int i);
int pop1();
int pop2();
```

אל תשכחו לבדוק את כל מקרי הקצה.

פתרון:

```
int[] a = new int[SIZE];
int top1 = -1;
int top2 = a.length;

void push1(int i)
{
  if (top1 + 1 == top2)
    ERROR overflow
  top1++;
  a[top1] = i;
}
```

```
void push2(int i)
{
  if (top1 + 1 == top2)
    ERROR overflow
  top2--;
  a[top2] = i;
}
int pop1()
{
  if (top1 == -1)
    ERROR underflow
  int x = a[top1];
  top1--;
  return x;
}
void pop2()
{
  if (top2 == a.length)
    ERROR underflow
  int x = a[top2];
  top2++;
  return x;
}
```

3. הראו כיצד ניתן לממש תור באמצעות שתי מחסניות. תארו כיצד יבוצעו $enqueue()$ ו־$dequeue()$ והסבירו את הסיבוכיות שלהם. נסו למצוא שיטות כך שבממוצע כל הכנסה והוצאה תהיה $O(1)$.

פתרון:

אנחנו נכניס אברים חדשים למחסנית A ונוציא אברים מהמחסנית B. אפשר לחשוב על ראש המחסנית B כתחילת התור ועל ראש המחסנית A כסופו.

במקרה שהמחסנית B ריקה ואנחנו רוצים להוציא אבר מהתור נהפוך את כל המחסנית A לתוך B. ונמשיך משם.

```
Stack A = new Stack();
Stack B = new Stack();

void enqueue(int i)
{
```

```
  A.push(i);
}
int dequeue()
{
  if (!B.isEmpty())
    return B.pop();
  if (A.isEmpty())
    ERROR underflow
  while (!A.isEmpty)
    B.push(A.pop());
  return B.pop();
}
```

סיבוכיות זמן הריצה של פעולה בודדת היא במקרה הגרוע $O(n)$ כי במקרה הגרוע, לפני הפעולה יש להפוך את כל המחסנית לתוך המחסנית השנייה. אבל בממוצע, כל אבר שנכניס יעבור 3 פעולות: הכנסה, העברה למחסנית השנייה והוצאה, כך שיוצא $O(1)$.

חישוב ביטוי מתמטי

1. הדגימו את ריצת האלגוריתם שלמדנו בכיתה לחישוב הביטוי

$$4 + (3 * 5 + 2 * 7) + (7 + 2) * 11$$

פתרון: קודם כל אנחנו הופכים את הביטוי ל reverse polish notation. נתאר

את הפלט ואת המחסנית בכל שלב:

Token	*Stack*	*Output*
4		4
+	+	4
(	+(	4
3	+(	4 3
*	+(*	4 3
5	+(*	4 3 5
+	+(+	4 3 5 *
2	+(+	4 3 5 * 2
*	+(+*	4 3 5 * 2
7	+(+*	4 3 5 * 2 7
)	+	4 3 5 * 2 7 * +
+	+	4 3 5 * 2 7 * + +
(	+(	4 3 5 * 2 7 * + +
7	+(	4 3 5 * 2 7 * + + 7
+	+(+	4 3 5 * 2 7 * + + 7
2	+(+	4 3 5 * 2 7 * + + 7 2
)	+	4 3 5 * 2 7 * + + 7 2 +
*	+*	4 3 5 * 2 7 * + + 7 2 +
11	+*	4 3 5 * 2 7 * + + 7 2 + 11
		4 3 5 * 2 7 * + + 7 2 + 11 * +

ואז נחשב מביטוי זה את הערך שלו. שוב בעזרת מחסנית:

Token	*Stack*
4	4
3	4, 3
5	4, 3, 5
*	4, 15
2	4, 15, 2
7	4, 15, 2, 7
*	4, 15, 14
+	4, 29
+	33
7	33, 7
2	33, 7, 2
+	33, 9
11	33, 9, 11
*	33, 99
+	132

ולכן התוצאה היא 132.

2. כתבו אלגוריתם הקורא ביטוי מתמטי ובודק האם הוא חוקי. למשל הביטוי הבא הוא חוקי:

$$3 + ((1 * 10 + 2) * 3) * 131$$

הדרכה: הכניסו את הקלט למחסנית. חישבו איך המחסנית צריכה להיראות ברגע שאתם רואים את הסוגר הראשון. אל תישכחו לחשוב איך המחסנית צריכה להראות כאשר נגמר כל הקלט. לשם בדיקת האלגוריתם שלכם כדאי לכם להריץ אותו על הביטוי לדוגמא.

פתרון:

כשאנו רואים את הסוגר הראשון, עד לפותח הראשון במחסנית חייב להיות משהו מהצורה:

$$(5 + 3 * 15/4 + 4$$

אז נוציא מהמחסנית ונבדוק שזה בדיוק משהו מהצורה הזאת. אחרי שמחקנו את כל זה, נכניס במקום כל זה מספר סתם - נגיד 0 - כי סוגריים שלמים בסופו של דבר עומדים במקום מספר בודד.

```
boolean checkExpression()
{
  Stack s = new Stack();
  while(t = nextToken()) {
    if (t != ')')
      s.push(t);
    else {
      if (s.pop(t) is not int)
        return false;
      do {
        if (s.isEmpty() || s.pop(t) is not op)
          return false;
        if (s.isEmpty() || s.pop(t) is not int)
          return false;
      } while (s.peek() != '(');
      s.pop() // Takes out the '('
      s.push(0) // just a number
    }
  }
  if (s.isEmpty())
    return true;
  if (s.pop(t) is not int)
```

```
    return false;
  while (!s.isEmpty()) {
    if (s.pop(t) is not op)
      return false;
    if (s.isEmpty() || s.pop(t) is not int)
      return false;
  }
  return true;
}
```