מטלה מספר 3 – שפות תכנות והידור (Bison)

במטלה זו נרשום Interpreter לשפה משלנו המכילה **מספרים שלמים**, **מחרוזות** ו- Maps.
במטלה תתרגלו את השימוש ב- Flex ו- Bison. לפיכך, ככל שתכתבו פחות קוד ב- C ככה העבודה שלכם תהיה טובה יותר!

מונחים בסיסיים:

WS – רווח או **\t** (טאב) או **\n** (newline)

VARNAME – רצף של תווים המתחילים ב- **a-z A-Z _** וממשיכים באפס או יותר מהתווים **a-z A-Z 0-9 _**

טיפוסים

שלמים

רצף של ספרות (לפחות אחת).
לדוגמא: **13**.
אם אורך הרצף הוא יותר מ1, אזי הספרה הראשונה לא יכולה להיות **0**.
לדוגמא: **06** אינו מספר שלם תקין.
במקרה והמשתמש יזין מספר כזה, מערכת ה- Bison תוציא כברירת מחדל את ההודעה: "Error: syntax error" (אין צורך שתוציאו הודעה אחרת...)

עבור כל מספר שלם שנקליד ל- interpreter יודפס
Expr = <int_expr>

```
12      // OK
Expr = 12
0       // OK
Expr = 0
999     // OK
Expr = 999
```

אופרטורים:

סדר קדימות	תווים	תיאור	אסוציאטיביות
גבוהה ביותר	()	קיבוץ	
	^	חזקה	Right
	+ -	פעולות אונריות	
	* /	כפל / חילוק	left
	+ -	חיבור, חיסור	left
	== =!	השואה	
	&&	AND	left
נמוכה ביותר	||	OR	left

1. הפעולה ^ היא פעולת חזקה – אפשר להניח כי הבסיס הוא תמיד מספר חיובי (כולל 0).
2. עבור הפעולות הבוליאניות, נקבע כי ביטוי השונה מ- 0 הוא true , וביטוי השווה ל- 0 הינו false.

אם נבצע חלוקה ב- 0 תודפס ההודעה

```
Error – Zero Division // including \n
```

3. עבור פעולת / יש לבדוק כי אין חלוקה ב- 0. (הביטוי המתקבל הוא מספר שלם).

```
1 + 3 * 5^2
Expr = 76
-2 - 5
Expr = -7
-2 - - 5
Expr = 3
-2 - - -5
Expr = -7
-2 + + -5
Expr = -7
11 == 1 + 6
Expr = 0
11 == 5 + 6
Expr = 1
6 && 5
Expr = 1
6 && 5 && 0
Expr = 0
6 && 5 || 0
Expr = 1
-6 && (5 || 0)
Expr = 1
```

מחרוזות

רצף של תווי ASCII המוקפים ע"י גרשיים (").

```
"father"
"@#$ _ %% xyz"
```

אופרטורים:

סדר קדימות	תווים	תיאור	אסוציאטיביות
גבוהה ביותר	()	קיבוץ	
	<>	Int-To-String	
	^	היפוך רשימה	right
	[]	תת מחרוזת	
	+	שרשור	left
נמוכה ביותר	=~	חיפוש	

כאשר נקליד מחרוזת, יודפס

Str = <str_exp>

```
"a" + "b"
Str = ab
"a" + <2^2>
Str = a4
"a" + "b" + "c"
Str = abc
^"abc"
```

```
Str = cba
^"abc" + "d"
Str = cbad
^("abc" + "d")
Str = dcba
"abcdefg"[1..3]
Str = bcd
"abcdefg"[1]
Str = b
^"abc"[0]
Str = c
^("abc"[0])
Str = a
"abccde" =~ "c"
Expr = 2
"I like ice cream" =~ "ice"
Expr = 7
"abccde" =~ "f"
Expr = -1
2^"abcd"=~"d"
Expr = 8
2^2^"abcd"=~"d"
Expr = 256
```

1. פעולת התת מחרוזת יכול להתבצע על תו יחיד `"cactus"[1]` או על רצף של תווים `"cactus"[1..2]`
   אינדקס של תו יחיד יכול להיות **ביטוי חשבוני** `"cactus"[1-1^5]`
   אינדקסים של רצף תווים חייב להיות **מספרים שלמים**.
   במקרה ויש חריגה מהאינדקס יש להדפיס את השגיאה:

```
Error - Index out of range! // including \n
```

2. X =~ Y היא פעולה על מחרוזות המחזירה את האינדקס של המופע הראשון של Y ב- X. אם Y לא נמצא ב- X יוחזר 1-.
3. כאשר המשתמש יבצע פעולת שרשור בין מספר לבין מחרוזת (מבלי לבצע המרה למחרוזת) מערכת ה- Bison תוציא כברירת מחדל את ההודעה: "Error: syntax error" (אין צורך שתוציאו הודעה אחרת...)
4. הפעולה + בעלת סדר קדימויות גבוה יותר מ- X =~ Y.

```
"abc" =~ "b" + 1     // Syntax Error!
("abc" =~ "b") + 1   // = 2
```

<u>משתנים</u>

אנו יכולים להגדיר **משתנים קבועים** באופן הבא:

```
var <VARNAME> = expr
```

כאשר expr הוא ביטוי חשבוני **המחזיר מספר שלם**.

```
var x = 99        // x is Int
var y = 9 * 5     // y is Int
```

במקרה ומשתנה מוגדר פעמיים נדרוס את ערכו

```
var x = 99        // x is Int
var x = 100
x
Expr = 100
```

והשימוש במשתנים מתבצע באופן הבא:

```
var y = 100
var z = 10
y + 1
Expr = 101
```

```
100 + y
Expr = 200
y + z
Expr = 110
```

במקרה ונשתמש במשתנה שלא הוגדר נדפיס את ההודעה:

```
The variable %s doesn't exist!  // including \n
```

Map

אוסף של זוגות ערכים value : key כאשר:

key – הוא מחרוזת מהצורה **VARNAME**

value – הוא ביטוי חשבוני (מספר שלם)

כאשר נכתוב map אל ה- interpreter נדפיס את ה- keys והערכים המתאימים שלהם:

```
{} // empty map
{a:1, b:2, c:3}
a : 1
b : 2
c : 3
{fish: 1^5-99, x_1: (19 + 6) / 5, t2: "aa" =~ "b"}
fish : -98
x_1 : 5
t2 : -1
```

אם key מופיע פעמיים נדפיס את ההודעה:

```
The key %s already exist!  // including \n
```

כדי להשיג את ערכו של key מסוים נשתמש ב- [<keyName>]

```
{a:1, b:2, c:3}["b"]
Expr = 2
```

במקרה ו-key לא קיים ב- Map נדפיס את ההודעה:

```
The key %s doesn't exist!  // including \n
```

<u>הנחיות כלליות</u>

- המטלה צריכה לרוץ על שרתי ה- U2 של האוניברסיטה.
- הגשה דרך מערכת ההגשות
- יש לצרף בהגשה את הקבצים הבאים:
  - makefile
  - ex3.lex
  - ex3.y
- בראש הקבצים (ה- FLEX וה- BISON) יש להוסיף הערה בה רשום:
  - שם הסטודנט
  - ת.ז.
  - מספר קבוצה
  - שם משתמש
- קובץ ההרצה הסופי צריך להיות קרוי a.out.