תורת הקומפילציה

תרגיל בית 1 – בניית מנתח לקסיקלי באמצעות Flex

מתרגל אחראי לתרגיל: אדם בוטח

יש להפנות שאלות על התרגיל במייל ל- botach@campus.technion.ac.il עם "236360_" בתחילת כותרת ההודעה.

ההגשה ביחידים או בזוגות.

לתרגיל ייפתח דף FAQ באתר הקורס. כל הבהרה שתופיע בו עד יומיים ממועד ההגשה תהווה הוראה מחייבת.

הנחיות כלליות

- בתרגיל זה תממשו מנתח לקסיקלי עבור **וריאציה מצומצמת** של השפה **Swift**. השפה פותחה ע"י חברת Apple ושוחררה לראשונה בשנת 2014 בכדי להציע אלטרנטיבה מודרנית לשפה Objective-C כשפת הפיתוח המרכזית עבור מרבית הפלטפורמות של החברה (מכשירי iOS, iPadOS, MacOS וכו').
- הבהרה – הכרות עם השפה ו/או שימוש במחשב המריץ MacOS **אינם** נחוצים לצורך פתרון התרגיל.
- התרגיל ייבדק אוטומטית. **הקפידו למלא אחר ההוראות במדויק.** הבדיקה תתבצע על csComp.
- יש להשתמש ב-flex בלבד (ולא ב-lex).

הגדרת מושגים כלליים

- רווח לבן: אחד מבין: רווח (ספייס), טאב, CR (התו \r), LF (התו \n).
- תווים ניתנים להדפסה: התווים שערך ה-ascii שלהם בין 0x20 ל-0x7E, או רווחים לבנים: טאב (0x09), LF (0x0A), CR (0x0D) (רווח רגיל נכלל בתוך הטווח).
  - ניתן לקרוא על תווים ניתנים להדפסה בהרחבה בוויקיפדיה בערך הבא: https://en.wikipedia.org/wiki/ASCII#Printable_characters
- רצף מילוט (escape sequence): לוכסן אחורי (התו \) ואחריו תו או יותר שביחד מפורשים כתו אחר.
  - דוגמאות: \n – ירידת שורה, \t – טאב.
  - ניתן לקרוא על רצפי מילוט בהרחבה בוויקיפדיה בערך הבא: https://en.wikipedia.org/wiki/Escape_sequences_in_C
- רצף מילוט של תו ASCII[1]: רצף בעל התבנית הבאה: \u{n} כאשר n הינו מספר בייצוג הקסדצימלי (המורכב מהספרות 0-9 והאותיות a-f גדולות/קטנות) באורך 1 עד 6 תווים. הרצף מהווה מילוט לתו שערך ה-ASCII שלו בייצוג הקסדצימלי הינו n. דוגמא: \u{41} הינו רצף מילוט עבור התו **'A'**.
  הערה: בהמשך יוסבר אופן הטיפול בערך ascii שאינו חוקי או אינו ניתן להדפסה.

בטבלה בעמוד הבא מפורטים האסימונים (tokens) שאיתם נעבוד בתרגיל וההגדרות שלהם.

שימו לב – אלא אם כן נכתב במפורש אחרת כל הגדרות האסימונים הינן case-sensitive.

[1] בגירסא המקורית של Swift התבנית \u{n} משמשת לייצוג תווי Unicode. בתרגיל זה אנו נתמוך באופן חלקי בתווי ASCII בלבד.

<u>הגדרת האסימונים</u>

שם האסימון	תיאור	ערכים אפשריים	דוגמאות (לקסמות המתאימות לאסימון זה)
ID	שם של מזהה	מתחיל באות (גדולה או קטנה) <u>או</u> קו תחתון, ואחריה רצף של אותיות (גדולות או קטנות) או ספרות. שימו לב – אם התו הראשון הינו קו תחתון הרצף שמגיע אחריו <u>אינו ריק</u>.	a _compi _Z123ab _cC1aA
BIN_INT	ערך מספרי שלם <u>אי-שלילי</u> בבסיס בינארי (בסיס 2)	מספר המתחיל בתווים 0b ואחריהם רצף <u>לא ריק</u> המורכב מהספרות 0 או 1.	0b1010 0b0
OCT_INT	ערך מספרי שלם <u>אי-שלילי</u> בבסיס אוקטלי (בסיס 8)	מספר המתחיל בתווים 0o ואחריהם רצף <u>לא ריק</u> המורכב מהספרות 0-7.	0o700
DEC_INT	ערך מספרי שלם <u>אי-שלילי</u> בבסיס דצימלי (בסיס 10)	מספר המורכב מספרה דצימלית (<u>אחת או יותר</u>).	123 0 42
HEX_INT	ערך מספרי שלם <u>אי-שלילי</u> בבסיס הקסדצימלי (בסיס 16)	מספר המתחיל בתווים 0x ואחריהם רצף לא ריק של האותיות a-f (גדולות או קטנות) והספרות 0-9.	0x10 0x7FFFFFFF 0xFf
DEC_REAL	מספר ממשי <u>אי-שלילי</u> בבסיס דצימלי	אחד משני הבאים: 1. מספר המורכב מספרה דצימלית אחת או יותר, עם נקודה עשרונית אחת לפניו, אחריו או בתוכו. 2. מספר ממשי כמתואר בסעיף 1 ובתוספת אקספוננט בסופו. האקספוננט מורכב מהאות e (גדולה או קטנה), אחריה סימן + או – ואז מספר <u>שלם</u> בייצוג דצימלי.	0.0 .2 8. 42.125 1.e-8 1.E+10
HEX_FP	מספר ממשי <u>אי-שלילי</u> בבסיס הקסדצימלי בייצוג Floating Point	מספר המתחיל בתווים 0x, אחריהם רצף לא ריק המורכב מהאותיות a-f (גדולות או קטנות) והספרות 0-9, ואז אקספוננט המורכב מהאות p (גדולה או קטנה), אחריה סימן + או - ואז מספר שלם בייצוג <u>דצימלי</u>. **<u>דוגמא:</u>** הלקסמה 0xFp-2 מייצגת בבסיס דצימלי את הערך: $15 \times 2^{-2} = 3.75$	0xFp-2 0x6P+3
STRING	ליטרל המייצג מחרוזת	אוסף תווים ניתנים להדפסה באורך אפס או יותר, בתוך מרכאות כפולות, בשורה אחת. **הערות:** 1. צירוף האותיות \n יכול להופיע (כשני תווים, לוכסן אחורי ואחריו האות n) במחרוזת, אך ירידת השורה עצמה כתו אחד אינה יכולה להופיע. כנ"ל עבור \r. 2. המחרוזת אינה יכולה להכיל מרכאות כפולות " ולוכסן אחורי \ אלא אם הם חלק מרצף מילוט חוקי כמפורט בסעיף הבא.	"this is a \t legal string \n" "also legal \n\r\\" "we can \"quote\" inside!" "\u{54}\u{69}\u{4D}\u{65}" "\u{57}\u{61}\u{53}\u{74}\u{45}\u{64}"

		3. המחרוזת תומכת ברצפי מילוט באופן חלקי. נגביל את התווים שמותר לכתוב אחרי הלוכסן האחורי במחרוזת ונתמוך בששת המקרים הבאים בלבד: • \n • \r • \t • \\ • \" • \u{n} – רצף מילוט של תו ascii כפי שהוסבר לעיל. אופן הטיפול ברצפי המילוט יוסבר בהמשך, בחלק של הדפסת אסימונים. **שימו לב:** כל רצף מילוט שאינו ברשימה הנ"ל <u>אינו מהווה קלט חוקי</u>. 4. ניתן להניח שהאורך של מחרוזת בלי המרכאות בהתחלה ובסוף לא עולה על 1024 תווים.	
TYPE	שם של טיפוס בסיסי	אחד מן הבאים (שימו לב שהשמות הינם case sensitive): Int, UInt, Double, Float, Bool, String, Character	Int Double Float
VAR	משמש להצהרה על משתנה חדש	var	var
LET	משמש להצהרה על קבוע חדש	let	let
FUNC	משמש להגדרת פונקציה חדשה	func	func
IMPORT	משמש לייבוא ספריות	import	import
NIL		nil	nil
WHILE		while	while
IF		if	if
ELSE		else	else
RETURN	משמש לחזרה מפונקציה	return	return
SC	נקודה-פסיק	;	;
COMMA	פסיק	,	,
LPAREN	סוגר מעוגל שמאלי	(	(
RPAREN	סוגר מעוגל ימני	)	)
LBRACE	סוגר מסולסל שמאלי	{	{
RBRACE	סוגר מסולסל ימני	}	}
LBRACKET	סוגר מרובע שמאלי	[	[
RBRACKET	סוגר מרובע ימני	]	]
ASSIGN	מציין השמה	=	=

RELOP	אופרטורים רלטיביים	== | != | < | > | <= | >=	== >=
LOGOP	אופרטורים לוגיים	&& או ||	&& ||
BINOP	אופרטורים מתמטיים בינאריים	+ | - | * | / | %	+ -
TRUE	ערך לוגי 1	true	true
FALSE	ערך לוגי 0	false	false
ARROW	מציין ערך החזרה בהגדרת פונקציה	->	->
COLON	משמש לציון explicit של טיפוסים	:	:
COMMENT	הערות	אחד מהשניים הבאים: 1. מתחיל ב */ ומסתיים ב /*. בין הפותח והסוגר יכול להופיע כל תו שניתן להדפסה (כולל ירידות שורה) פרט לצירוף */. 2. מתחיל ב // ומסתיים בירידת שורה (CR, LF). לאחר הפותח יכול להופיע (כחלק מההערה) כל תו שניתן להדפסה לא כולל ירידות שורה (CR,LF).	/* this is a comment */ /* this is also a valid comment. Note that it is spread across multiple lines. */ // this is a single line comment

פעולת המנתח והפלט הנדרש

המנתח יתעלם מכל הרווחים הלבנים, חוץ מבתוך מחרוזות.

כאשר המנתח מזהה אסימון, יש לפלוט שורה בפורמט הבא (יש לדאוג לרווח יחיד בין כל רכיב שורה ולירידת שורה ע"י LF (\n) בלבד לאחר הרכיב האחרון):

<line number> <token name> <value>

כאשר:

- line number: מספר השורה בה האסימון **מסתיים**.
- token: שם האסימון שזוהה (לפי השמות בחלק "הגדרת אסימונים" למעלה).
- value: ערך האסימון שזוהה, כלומר הלקסמה, פרט למקרה של מחרוזות, מספרים שלמים והערות כפי שיוסבר בהמשך.

הדפסת הלקסמה של מחרוזות:

מחרוזות יודפסו ללא המרכאות הכפולות המקיפות אותן. בנוסף נרצה לטפל ברצפי המילוט באופן הבא:

- \n,\r,\t יוחלפו בסוג המתאים של רווח לבן (טאב, CR, LF).
- \\ יוחלף בלוכסן אחורי יחיד (\).
- רצף מילוט של תו ascii: אם הרצף (המיוצג כאמור ע"י n בתבנית \u{n}) מהווה ייצוג הקסדצימלי חוקי של תו שניתן להדפסה, אז יש להדפיס את התו המתאים במקום רצף המילוט. אחרת, יש להדפיס שגיאה כפי שיפורט בהמשך תחת 'טיפול בשגיאות'.

דוגמה – המחרוזת הבאה:

"\u{49}\u{20}\"L\u{6F}ve\"\tCom\u{00070}i!\nCompi Loves Me\u{21}"

תודפס בפורמט הנדרש באופן הבא:

1 STRING I "Love" Compi!
Compi Loves Me!

הדפסת הלקסמה של מספרים שלמים:

יש להדפיס את ערכם המספרי של מספרים שלמים (המיוצגים ע״י האסימונים: BIN_INT, OCT_INT, DEC_INT HEX_INT) כערך דצימלי (יש להדפיס ערכים חיוביים ללא סימן ה-'+').

עבור מספרים ממשיים (המיוצגים ע״י האסימונים DEC_REAL, HEX_FP) תודפס הלקסמה כפי שהופיעה בקלט.

הדפסת הלקסמה של הערות:

במקום תוכן ההערה, יודפס מספר השורות בהערה, כלומר מספר המופעים של ירידות שורה, פלוס 1. **זכרו:** ירידת שורה עשויה להיות אחד הרצפים הבאים: CRLF (\r\n), CR (\r), LF (\n). הרצף \r\n נחשב לירידת שורה אחת.

שימו לב שההוראות האלה מתייחסות להערות משני הסוגים שפורטו לעיל.

דוגמה

עבור הקלט:

```
var x : String = "\u{54}o \u{6C}nf\u{69}n\u{69}ty, a\u{6E}d B\u{65}y\u{6F}nd!"

while true {

    print(x)

}

// something important
```

פלט המנתח יהיה:

```
1 VAR var

1 ID x

1 COLON :

1 TYPE String

1 ASSIGN =

1 STRING To Infinity, and Beyond!
```

2 WHILE while

2 TRUE true

2 LBRACE {

3 ID print

3 LPAREN (

3 ID x

3 RPAREN)

4 RBRACE }

5 COMMENT 1

<u>הערות נוספות על תווים בקובץ</u>

ניתן להניח כי קבצי הדוגמאות הם קבצי ascii בלבד (כלומר: אינם UTF-8 או UTF-16). בהכינכם קבצי בדיקה, וודאו כי אתם מכוונים את ה-Encoding של הקובץ ל-ASCII או ANSI, או מבצעים save as כ-ASCII. לנוחותכם, וכדי למנוע בעיות בהעתקה בין קבצים, להלן מפתח של התווים המוזכרים בתרגיל וערכי ה-ASCII שלהם:

שם	סימן	ערך ASCII (hex)
סוגר מסולסל שמאלי	{	7B
סוגר מסולסל ימני	}	7D
נקודתיים	:	3A
שווה	=	3D
לוכסן אחורי	\	5C
נקודה פסיק	;	3B
מינוס / מקף	-	2D
פלוס	+	2B
פסיק	,	2C
קו תחתון	_	5F
נקודה	.	2E
מרכאות כפולות	"	22
Carriage return	CR	0D
Line feed	LF	0A
רווח		20
טאב		09
סוגר משולש שמאלי	<	3C
סוגר משולש ימני	>	3E
כוכבית	*	2A
לוכסן (סלש)	/	2F
אחוז	%	25
אמפרסנד	&	26
קו ניצב	|	7C

קבצי הטסט זמינים בקובץ zip ומומלץ תמיד להוריד ולהעביר אותם כ-zip על מנת למנוע שינוי אוטומטי של ירידות השורה על ידי תכנות להעברת קבצים.

<u>טיפול בשגיאות</u>

הערה: אחרי הדפסת ההודעה המתאימה לשגיאה <u>הראשונה</u> בה נתקלתם, יש **<u>לסיים את התכנית</u>** (לשם כך היעזרו בפקודה exit(0)).

1. כאשר המנתח נתקל בתו לא חוקי יש להדפיס:

Error <*char*>\n

כך שעבור הקלט הבא:

```
@
```

הודעת השגיאה תהיה:

```
Error @
```

(רווח בודד בין Error ל @, וירידת שורה בסוף השורה על ידי LF בלבד.)

2. כאשר <u>שורה</u> מסתיימת באמצע מחרוזת, יש להדפיס:

Error unclosed string\n

3. כאשר הקלט מסתיים באמצע הערה (מהסוג הראשון המופיע בטבלת האסימונים), יש להדפיס:

Error unclosed comment\n

4. כאשר מחרוזת מכילה רצף מילוט שלא מופיע בהגדרת התרגיל, יש להדפיס:

Error undefined escape sequence <*sequence*>\n

כאשר <sequence> הינו התו הראשון ברצף המופיע לאחר הלוכסן האחורי. <u>לדוגמא</u>:

עבור מחרוזת המכילה את הרצף הלא חוקי: **`\q`** , הודעת השגיאה תהיה:

```
Error undefined escape sequence q
```

עבור מחרוזת המכילה את הרצף הלא חוקי: **`\u{@@@}`** , הודעת השגיאה תהיה:

```
Error undefined escape sequence u
```

5. כאשר **בתוך הערה** נקרא הרצף ***/**, יש להדפיס:

Warning nested comment\n

שימו לב: אין להדפיס את האסימון של COMMENT במקרה זה, אלא יש כאמור לסיים את התכנית מיידית.
הערה: מטרת בדיקה זו היא להימנע מטעות של מתכנתים רבים: אין לקנן הערות, ברוב שפות התכנות.
הנה דוגמה בעייתית בשפת C:

```
int nest = /* /*/ 0 */* */ 1; // our guess:     int nest = 1;
int nest = /* /*/ 0 */* */ 1; // actual result: int nest = 0 * 1;
```

הוראות הגשה

עליכם להגיש קובץ zip המכיל קובץ אחד בלבד בשם hw1.lex דרך אתר הקורס.

דרישות נוספות והערות

על המנתח להיבנות על השרת csComp בעזרת הפקודות הבאות:

```
flex hw1.lex
gcc -ll lex.yy.c
```

מנתח שלא יבנה בהצלחה בעזרת הפקודות הללו **יקבל 0 אוטומטית**.

בתרגיל זה (כמו בתרגילים אחרים בקורס) ייבדקו העתקות. אנא כתבו את הקוד שלכם בעצמכם.

שימו לב: מומלץ (מאוד) להיוועץ ב-manual של flex לצורך ביצוע התרגיל. קל יותר לבצע אותו על ידי שימוש ביכולות מתקדמות של flex שלא נלמדו בתרגולים, כגון **start conditions**, regex patterns מתקדמים, ו-debug mode.

טיפ: תוכלו להשתמש באתר https://regexr.com שעוזר בהבנה ובבנייה של תבניות regex מורכבות.

בדיקת המנתח

באתר הקורס מופיע קובץ zip המכיל קבצי בדיקה לדוגמה t1.in, t1.out ו-t2.in, t2.out

ניתן ואף רצוי לבדוק את עצמכם באופן הבא:

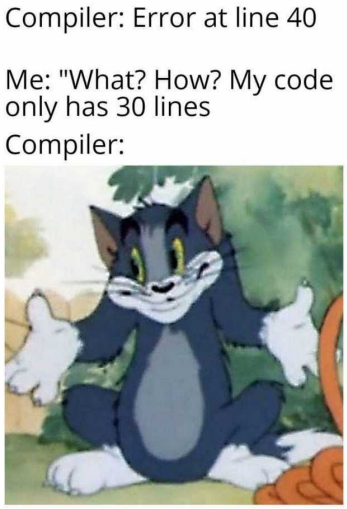
בנו את המנתח על ידי הפקודות לעיל על השרת csComp. העבירו את קובץ ה-zip של הקבצים לדוגמא לשרת ובצעו unzip. נניח שקובץ ההרצה של המנתח הוא a.out, אזי יש להריץ:

```
./a.out < t1.in >& t1.res
diff t1.res t1.out
```

ולבדוק שמתקבל diff ריק.

שימו לב - במידה והמנתח שלכם לא עובר את כל קבצי הבדיקה שסופקו מראש לא תתאפשר הגשה חוזרת של התרגיל. בכדי למנוע מצבים כאלה - באתר מופיע script לבדיקה עצמית לפני הגשה בשם selfcheck-hw1 בו אתם נדרשים להשתמש בו על מנת לוודא את תקינות ההגשה שלכם. את הסקריפט ניתן להריץ על השרת csComp בלבד באמצעות הפקודה:

```
./selfcheck-hw1 hw1.zip
```



בהצלחה!