תורת הקומפילציה

תרגיל בית 1 – שימוש ב-Flex

מתרגל אחראי לתרגיל: אנטוניו אבו נסאר

ההגשה בזוגות.

יש להפנות שאלות על התרגיל במייל ל-antonio@cs בלבד.

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

הנחיות כלליות

- בתרגיל זה תממשנה מנתח לקסיקלי שיוכל לטפל בווריאציה של פורמט **CSS** – גיליונות סגנון מדורגים – שמשמש לעיצוב דפי אינטרנט.
- התרגיל ייבדק אוטומטית. **הקפידו למלא אחר ההוראות במדויק.** הבדיקה תתבצע על csl3.
- יש להשתמש ב-flex בלבד (ולא ב-lex).

דוגמה לקוד ב-CSS

```
table ~ div#first img:hover, table.colorful > input[type="text"]
{
    text-align: center;
    background: white url('../Images/BehindBG2.png') fixed center;
    background-size: 100% 90%;
    width: 62.5em;
    color: rgb(197, 90, 17);
}
```

הסבר בקצרה והגדרת מושגים בשפה

אין צורך בהבנת הקוד, וניתן לפתור את כל התרגיל בלי לקרוא את ההסבר, אך הדוגמה מובאת למען הגדרת מספר מושגים, כדי ליצור שפה משותפת למשך התרגיל שתפשט את ההסברים.

הקוד הנ"ל מהווה דוגמה ל**כלל סגנון** עבור דף אינטרנט. בקובץ CSS לרוב ישנם מספר כללי סגנון המשורשרים אחד אחרי השני, ואינם יכולים להיות מקוננים.

בתחילת כלל סגנון (בשורה הראשונה בדוגמה) נמצאת **רשימת סלקטורים** (selector list) המופרדים בפסיקים, כאשר כל סלקטור מהווה "תבנית" (pattern) לאיברים המרכיבים את דף האינטרנט, כך שעל כל איבר המתאים לתבנית זו מופעל כלל סגנון זה. בדוגמא לעיל יש שני סלקטורים:

```
table ~ div#first img:hover (1)        table.colorful > input[type="text"] (2)
```

בסלקטור (2), הסוגריים המרובעות מגדירות **תכונת-תג**, המציינת שאנו מחפשים איברים מסוג input שקיימת אצלם תכונה בשם type עם ערך text.

בתוך כלל סגנון (כלומר בתוך הסוגריים המסולסלים), נמצאת רשימת **ההצהרות**. הצהרה מורכבת משם של **תכונה** (property) שאחריה נקודתיים ואז **ביטוי** (ערך - expression) הניתן לתכונה זו.

למי שמעוניין בהבנה נוספת (שאינה הכרחית לצורך התרגיל): ננסה להבין את הסלקטור (1) בדוגמה. נקרא אותו מימין לשמאל (כאשר מפרידים על ידי רווח). הסלקטור תופס את כל התמונות (`img`) שהעכבר מונח עליהן (`:hover`) שהן צאצאים (כלומר מוכלות בתוך) של חתך (`div`) עם מזהה בשם first (`#first`), שהוא בעצמו "אח" (`~`) של טבלה (`table`) כלשהי (כלומר לשניהם יש "הורה" משותף).

הגדרת מושגים כלליים

- <u>רווח לבן:</u> אחד מבין: רווח (ספייס), טאב, CR (התו \r), LF (התו \n).
- <u>נוטציה הקסדסימלית של מספר שלם:</u> מחרוזת המתחילה ב-0x או בסימן (+-) ואחריו 0x, ואחריהם המספר בייצוג הקסדסימלי (בבסיס 16). החלק של המספר שאחרי ה-0x חייב להכיל ספרה אחת לפחות, ועשוי להתחיל ברצף אפסים מובילים. אין לשים סימן בחלק שאחרי ה-0x.
  - <u>תזכורת:</u> ספרה הקסדסימלית היא מספר בין 0 ל- 9 או אות באנגלית (קטנה או גדולה) בין a ל- f.
  - <u>דוגמאות:</u> +0x0, 0xFF, +0x12bA, 0x0002, -0xfefe, -0x16.
- <u>תווים ניתנים להדפסה:</u> 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
- <u>תווים חוקיים לשם מזהה:</u> אותיות באנגלית (קטנות או גדולות), מספרים, מקפים או קווים תחתונים.
- <u>רצף בריחה (escape sequence):</u> לוכסן אחורי (התו \) ואחריו תו או יותר שביחד מפורשים כתו אחר.
  - <u>דוגמאות:</u> \n – ירידת שורה, \t – טאב.
  - ניתן לקרוא על רצפי בריחה בהרחבה בוויקיפדיה בערך הבא: https://en.wikipedia.org/wiki/Escape_sequences_in_C
- <u>רצף בריחה של תו ascii:</u> מתחיל בלוכסן אחורי ואחריו רצף של ספרות הקסדסימליות באורך 1 עד 6 תווים. מהווה רצף בריחה לתו שערך ה-ascii שלו בייצוג הקסדסימלי הינו הערך הכתוב אחרי הלוכסן.

הערה: בהמשך יוסבר אופן הטיפול בערך ascii שאינו חוקי או אינו ניתן להדפסה.

הגדרת האסימונים

שם האסימון	תיאור	ערכים אפשריים	דוגמאות (לקסמות המתאימות לאסימון זה)	אנטי-דוגמאות (לקסמות שאינן מתאימות לאסימון זה)
COMMENT	הערה מרובת-שורות (כמו ב-C).	מתחילה ב-**/*** ומסתיימת ב-***/**. **הערה:** בין הפותח והסוגר יכול להופיע כל תו שניתן להדפסה, <u>פרט</u> לצירוף ***/**. ראה הסבר לסיבה לכך בחלק של טיפול בשגיאות בהמשך.	/* this is a multiline comment */ /*so*is*this*/	/* a /* b */ c */ /*/
NAME	מילה בודדת או שם מזהה	רצף לא ריק של <u>אותיות חוקיות לשם מזהה</u> (ראה הגדרה למעלה), **עם ההגבלה** שהרצף מתחיל באות באנגלית, או במקף <u>יחיד</u> ואחריו אות באנגלית.	helloWorld -dash_underscore -dashing----thru11-	--two-dashes _internal 1st
HASHID	אחריה בא שם מזהה לאיבר בדף האינטרנט, או צבע בקידוד הקסדסימלי.	סולמית (#) ואחריה רצף לא ריק של <u>אותיות חוקיות לשם מזהה</u>, **עם ההגבלה** שהרצף מתחיל באות באנגלית, או **<u>במספר</u>**, או במקף <u>יחיד</u> ואחריו אות באנגלית. **שימו לב** להבדל בין הגבלה זו לבין הגבלת האסימון הקודם.	#deadbeef #123456_hello #-There #124604	#--two-dashes #_internal #-123456 #1.1
IMPORT	משמש לייבוא כללי סגנון מגיליונות סגנון אחרים.	@import **שימו לב:** האותיות חייבות להיות קטנות, ואין רווח אחרי ה-@ או בין אותיות המילה.	@import	@ import @IMPORT

IMPORTANT	בא בסוף של הצהרה, נותן לה עדיפות.	סימן קריאה ואחריו המילה important. **שימו לב:** המילה important יכולה להיות מורכבת מכל צירוף של אותיות גדולות וקטנות. **שימו לב:** רצף של רווחים לבנים יכול להופיע בין סימן הקריאה למילה, אך לא באמצע המילה.	!important ! iMpOrTaNt ! *<שורה חדשה>* important	!thisisimportant !_important !import ant ! /*hi*/ important
STRING	מחרוזת שורה אחת (שורה אחת מבחינת קובץ הקוד).	אוסף תווים ניתנים להדפסה באורך אפס או יותר, בתוך מרכאות מאותו סוג (מרכאות כפולות או בודדות), 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, וכך גם \r. 2. המחרוזת אינה יכולה להכיל מרכאות מהסוג המגדיר אותה (למשל אם המחרוזת במרכאות בודדות, לא ניתן לכתוב מרכאות בודדות כאות בתוך המחרוזת). 3. המחרוזת תומכת ברצפי בריחה (escape sequences) באופן חלקי. נגביל את התווים שמותר לכתוב אחרי הלוכסן האחורי במחרוזת ונתמוך בחמשת המקרים הבאים בלבד: • \n • \r • \t • \\ • רצף בריחה של תו ascii, כמו שמוסבר למעלה. רצף הקסדסימלי מסתיים כשהאורך מגיע ל-6 או כשנתקלים באות לא הקסדסימלית. אופן הטיפול ברצפי הבריחה יוסבר בהמשך, בחלק של הדפסת האסימונים. **שימו לב:** כל רצף בריחה שאינו ברשימה הנ"ל אינו מהווה קלט חוקי. 4. ניתן להניח שהאורך של מחרוזת בלי המרכאות לא עולה על 1024 תווים.	"simple" 'also simple' "escape new lines\n" '"hex"\0045a ' '\011111HELLO' "\01 'h'ello"	'unmatching" "unclosed "multi-lined string" "incep- " -tion" "bad escape \" here"
COMB	קיצור של המילה combinator. משמשים בסלקטור כדי לבחור בן ישיר (>), אח ישיר (+) ואח כללי (~).	> + ~	>| + ~	

COLON	מפריד בין שם התכונה והביטוי שלה.	:	:	
SEMICOLON	מופיע בסוף כל הצהרה ובסוף שורת import.	;	;	
LBRACE	פותח כלל סגנון.	{	{	
RBRACE	סוגר כלל סגנון.	}	}	
LBRACKET	פותח תכונת-תג.	[	[	
RBRACKET	סוגר תכונת-תג.	]	]	
EQUAL	בדיקת שוויון בתכונת-תג.	=	=	
ASTERISK	תבנית שתופסת את כל איברי דף האינטרנט.	*	*	
DOT	שם מחלקה של איברים בדף האינטרנט.	.	.	
NUMBER	ערך מספרי שלם.	מספרים שלמים בייצוג הדצימלי הרגיל או בייצוג הקסדסימלי. מספרים חייבים להכיל ספרה אחת לפחות. עשוי להופיע סימן.	+0 -0x1aF 1337	+ --10 1.1 0x 0x-F
UNIT	ערך מספרי שברי עם יחידות.	הערך המספרי הינו דצימלי, חסר סימן, ויכול להיות או שלם או בצורת שבר עשרוני. שבר עשרוני מכיל נקודה עשרונית <u>אחת</u> ואחריה לפחות ספרה אחת. לא חייב להופיע לפניה מספר (זה ייחשב כמו 0). היחידות חייבות להופיע והן רצף לא ריק של אותיות <u>קטנות</u> באנגלית המוצמדות למספר (בלי רווח לבן ביניהם), <u>או</u> הסימן % בלבד.	0.1px .15em 10cm 300.100seconds 15%	0.cm 1.1.1% 15sec% 20 +15cm
RGB	מייצג צבע במודל ה-RGB.	האותיות <u>הקטנות</u> rgb ואחריהן שלושה מספרים שלמים בתוך סוגריים ומופרדים בפסיקים. רווחים לבנים יכולים להופיע בין המספרים לפסיקים ובין המספרים לסוגריים, אך לא לפני הסוגריים.	rgb(15, 20, 30) rgb(015, 0, -514) rgb(1 , 2 , 03)	rgb (10,10,10) rgb(10.5,10,10) rgb(1, ,)

<u>פעולת המנתח והפלט הנדרש</u>

המנתח יתעלם מכל הרווחים הלבנים, חוץ מבתוך מחרוזות, ופרט לדרישה שאין רווחים לבנים בין @ לבין המילה import באסימון האחרון בטבלה.

ניתן להניח שכל הערכים המספריים בתרגיל ניתנים לאחסון על ידי הטיפוס int.

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 הערות ומחרוזות, כמוסבר להלן.

הדפסת הלקסמה של מחרוזות:

מחרוזות יודפסו ללא המרכאות הבודדות/הכפולות המקיפות אותן.

בנוסף, אם המחרוזת מוקפת במרכאות בודדות, אז נרצה לטפל ברצפי הבריחה באופן הבא:

- \n,\r,\t מוחלפים בסוג המתאים של רווח לבן (טאב, CR, LF).
- \\ מוחלפת בלוכסן אחורי יחיד (\).
- רצף בריחה של תו ascii: אם הרצף מהווה ייצוג הקסדסימלי של תו שניתן להדפסה, אז יש להדפיס את התו המתאים במקום רצף הבריחה. אחרת, אין להדפיס את רצף הבריחה, או שום דבר במקומו.

דוגמה – המחרוזת הבאה:

‘Hello **\57**orld!\r\nThis\tis\t**\63**oo**\00006C**, as a**\C0a**lways.`

תודפס בפורמט הנדרש באופן הבא (רצף הבריחה האחרון אינו תו ניתן להדפסה ולכן לא מדפיסים כלום במקומו):

1 STRING Hello **W**orld!
This is **c**oo**l**, as always.

לעומת זאת, אם אותה מחרוזת הייתה מוקפת במרכאות כפולות, היה מודפס:

1 STRING Hello \57orld!\r\nThis\tis\t\63oo\00006C, as a\C0always.

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.

דוגמה

עבור הקלט:

border: 1px ‘soli\64’ #934848; /* lovely */

פלט המנתח יהיה:

1 NAME border

1 COLON :

1 UNIT 1px

1 STRING solid

1 HASHID #934848

1 SEMICOLON ;

1 COMMENT 1

הערות נוספות על תווים בקובץ

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 מרובע שמאלי	]	5B
סוגר מרובע ימני	[	5D
סוגר מסולסל שמאלי	}	7B
סוגר מסולסל ימני	{	7D
נקודתיים	:	3A
שווה	=	3D
סימן קריאה	!	21
לוכסן אחורי	\	5C
סולמית	#	23
נקודה פסיק	;	3B
מינוס / מקף	-	2D
פלוס	+	2B
פסיק	,	2C
קו תחתון	_	5F
נקודה	.	2E
גרש	'	27
מרכאות כפולות	"	22
Carriage return	CR	0D
Line feed	LF	0A
רווח		20
טאב		09
שטרודל	@	40
סוגר משולש ימני	>	3E
טילדה	~	7E
כוכבית	*	2A
לוכסן (סלש)	/	2F

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.

טיפול בשגיאות

הערה: אחרי הדפסת ההודעה המתאימה לשגיאה הראשונה בה נתקלתם, יש לסיים את התכנית (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.)

שימו לב: סימן קריאה בלי המילה important, שטרודל (@) בלי המילה import, ואף % בלי מספר לפני, נחשבים כתווים לא חוקיים, ועל כן יש צורך להדפסה זו עבור מקרים אלה.

2. כאשר שורה מסתיימת באמצע מחרוזת, יש להדפיס:

Error unclosed string\n

3. כאשר הקלט מסתיים באמצע הערה, יש להדפיס:

Error unclosed comment\n

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

Error undefined escape sequence <*sequence*>\n

כך שעבור מחרוזת המכילה את הרצף **\q00** , הודעת השגיאה תהיה:

```
Error undefined escape sequence q
```

5. כאשר לפונקציה rgb() הועברו פרמטרים שגויים (במספרם או במבניהם), יש להדפיס:

Error in rgb parameters\n

דוגמאות למצב כזה: rgb(10), rgb(1.1,10,10), rgb(1,1,1,1), rgb(,2,3), rgb(), rgb(name).

6. 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 csl2 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 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 http://regexp.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 csl3. 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 כי במידה והמנתח שלכם לא עובר את כל קבצי הבדיקה שסופקו מראש, לא תתאפשר הגשה חוזרת של התרגיל.

שימו לב כי באתר מופיע script לבדיקה עצמית לפני ההגשה בשם selfcheck. תוכלו להשתמש בו על מנת לוודא כי ההגשה שלכם תקינה.

בהצלחה!