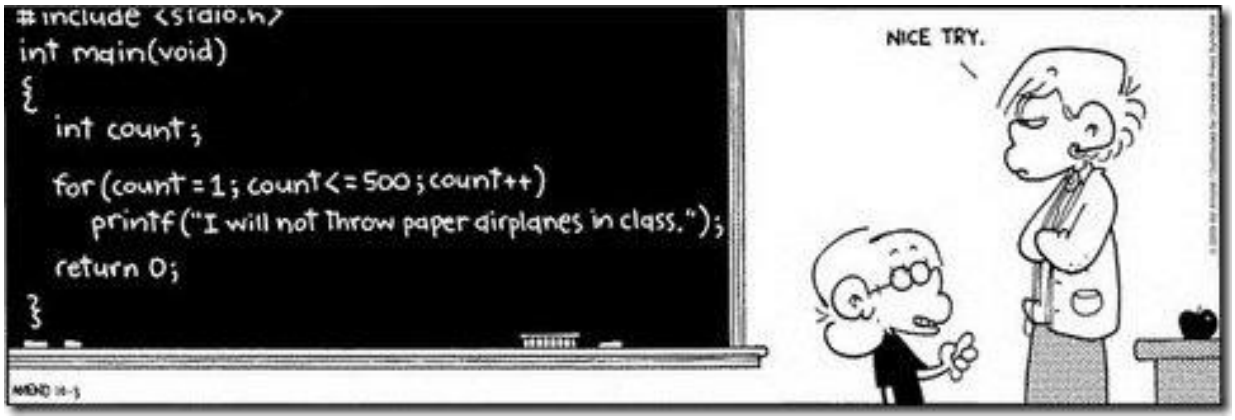
חוברת תרגילים קומפילציה

חוברה ע"י אסתי שטיין



מנתח לקסיקלי

שאלה 1

השפה הבאה Calc משמשת לביצוע חישובים מתמטיים. השפה מאפשרת להגדיר משתנים ,קבועים ופונקציות ,ותומכת במבני בקרה פשוטים ובקבלת ערכי המשתנים מהמשתמש.
בשפה מוגדרים האסימונים הבאים:

מלים שמורות:

שם האסימון	תיאור
TYPE	int, real, complex, pair
CONST	const
PRINT	print
INPUT	input
WHILE	while
FOR	for
FOREACH	foreach
IN	in
IF	if
THEN	then
ELSE	else
FUNCTION	funciton

קבועים מספריים ומחרוזות:

שם האסימון	תיאור	דוגמאות
INT	מספר שלם ע"פ הכללים הבאים: בלי אפסים מובילים, סימן אופציונאלי ואחריו קידומת '0d' אופציונאלית	10, 9+, 0d17,
HEX	מספר הקסדצימאלי עם קידומת 0x	0x14ad
REAL	מספר ממשי עם סימן אופציונאלי שחייב להכיל לפחות נקודה דצימלית (לא בהכרח אחרי ספרות, אבל אחריה חייבות להופיע ספרות) אקספוננט עם סימן אופציונאלי	13.45, -.123, 9e-12, -4.5e+55 +15.0
String	מחרוזת כלשהי מוקפת במרכאות	"lexer blabla23"

מבנים מורכבים: (שאסור שיופיע בתוכם white space)

PAIR	סוג סדור של מספרים. סוגריים משולשים וביניהם שני מספרים מופרדים בפסיק.	<2,5>, <-20,5.83>
COMPLEX	רכיב ממשי אופציונאלי ,ורכיב דמיוני (לא אופציונאלי) צמוד ל- i, מופרדים באופרטורים -/+ לא חשוב הסדר	1+2i, 6.18i-3, -3i
BIN_ARITH_OP	+, -, *, /, ^	
UN_ARITH_OP	++, --	
BIN_REL_OP	== (פעמיים שווה), =!, < , >	
ASSIGN	=	
LOGIC_OP	&&, ||	
NOT	!	

שונות:

IDENTIFIER	מחרוזת המורכבת מאותיות קטנות, גדולות, קו תחתון או מספרים. חייבת להתחיל באות.	variable,x34,s_2
SEP	;	
START_BLOCK	{	
END_BLOCK	}	
L_PARENT	(	
R_PARENT	)	
L_BRACKET	[	
R_BRACKET	]	
COMMA	,	

הערות:

תכנית בשפה CALC יכולה להכיל הערות בסגנון C++, כגון:

```
// This is a comment to end of line
/* This comment can take few lines */
```

שימו לב, שהערות **אינן** יכולות להיות מקוננות (הערה בתוך הערה). למשל:

```
/* This comment is /* */  45
```

ההערה בשורה הקודמת מסתיימת, והמספר 45 יפורש כINT.

המנתח צריך להתייחס אל white spaces כאל מפרידים ולא יבצע עבורם אף פעולה, והם לא יופיעו בפלט.

דרישות:

עליכם לכתוב מנתח לקסיקלי , לשפת Calc באמצעות Lex. המנתח צריך להדפיס עבור כל אסימון שזוהה את שם האסימון, מספר שורת הקלט, והערך שזוהה (lexeme), באופן הבא:

```
Line %d: Found token %s (lexeme: '%s').
Line 11: Found token HEX (lexeme: '0xabcd').
```

עבור תוים לא-מזוהים יש להדפיס את הודעת השגיאה הבאה (ולהמשיך בניתוח):

```
Line %d: Invalid token ('%s').
Line 11: Invalid token ('@').
```

פיתרון:

```
%option lex-compat
%option noyywrap
%option yylineno

%{
/* Declarations section */
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
%}
AAA                 [/]
BBB                 [*]
CCC                 [^/*]
LC     "/*"
RC     "*/"
P  "*"
S  "/"
A  "!"
B  "@"
C  "~"
```

```
D "`"
E "#"
F "$"
G "%"
H "&"
I "?"
J "`"
K "~"
L ""\"
M "|"
MM {M}|{L}|{UNDERLINE}|{SEP}

ZERO "0"
DOT "."
WHITESPACE  ([\t\n ])
LETTER [a-zA-Z]
SPACE " "
PLUS "+"
MINUS "-"
SIGN ({PLUS}|{MINUS})
NUM [0-9]
H_T [A-F]
H_TT [a-f]
NOT_Z [1-9]
PREINT "0d"
PREHEX "0x"

H_TTT  {NUMH_NOT_Z}{NUMH}
NUMH   ({NUM}|{H_T}|{H_TT})+
NUMH_NOT_Z ({NOT_Z}|{H_T}|{H_TT})+
HEX {PREHEX}({H_TTT})
TYPE {INT}|{REAL}|{COMPLEX}|{PAIR}
SIG1
{SPACE}|{DOT}|{MM}|{A}|{B}|{MORE_THEN}|{C}|{K}|{J}|{LESS_THEN}|{ADD}|
{SUB}|{D}|{E}|{F}|{I}|{G}|{XOR}|{H}|{P}|{L_PARENT}|{R_PARENT}|{L_BRAC
KET}|{R_BRACKET}



COMMENT_VAL
({LETTER}|{NUM}|{SIG1}|{START_BLOCK}|{PAIR}|{END_BLOCK}|{COMMA})
INT ({PREINT}?{ZERO})|({SIGN}?{PREINT}?{NOT_Z}{NUM}*)

T_NUM
(({NOT_Z}{NUM}*({DOT}({NUM}*{NOT_Z})+)?)|(({NOT_Z}{NUM}*)?{DOT}({NUM}
*{NOT_Z})+))

U_REAL (({T_NUM}"e"{SIGN}{NOT_Z}{NUM}*)|({T_NUM}))
REAL {SIGN}?{U_REAL}
COMPLEX ({REAL}"i"({SIGN}{U_REAL})?)|(({REAL}{SIGN})?{U_REAL}"i")

STRING "\""([^"])*"\""
PAIR "<"{REAL}","{REAL}">"
BIN_ARITH_OP {SPACE}({ADD}|{SUB}|{DIV}|{MUL}|{XOR}){SPACE}
ADD   "+"
SUB   "-"
DIV   "/"
MUL   "*"
XOR   "^"

UN_ARITH_OP {INC}|{DEC}
```

```
INC "++"
DEC "--"
BIN_REL_OP {MORE_THEN}|{LESS_THEN}|{NOT_EQUAL}|{EQUAL}
MORE_THEN   ">"
LESS_THEN   "<"
NOT_EQUAL   "!="
EQUAL       "=="
ASSIGN            "="
LOGIC_OP    {AND}|{OR}
AND         "&&"
OR                "||"
NOT               "!"
UNDERLINE   "_"
IDENTIFIER  {LETTER}({LETTER}|{NUM}|{UNDERLINE})*
SEP ";"
START_BLOCK "{"
END_BLOCK   "}"
L_PARENT    "("
R_PARENT    ")"
L_BRACKET   "["
R_BRACKET   "]"
COMMA       ","
ONE_LINE_COMMENT
"//"({LETTER}|{REAL}|{SPACE}|(".")|{START_BLOCK}|{PAIR}|{END_BLOCK}|{
L_PARENT}|{R_PARENT}|{COMMA})*
COMMENT   {AAA}({AAA}.*|{BBB}({AAA}|{CCC}|{BBB}+{CCC})*{BBB}+{AAA})


CONST "const"
PRINT "print"
INPUT "input"
WHILE "while"
FOR "for"
FOREACH "foreach"
IN "in"
IF "if"
THEN "then"
ELSE "else"
FUNCTION "function"
%%
{INT}             printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"INT",yytext);
{HEX}             printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"HEX",yytext);
{REAL}                  printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"REAL",yytext);
{PRINT}                 printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"PRINT",yytext);
{CONST}                 printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"CONST",yytext);
{INPUT}                 printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"INPUT",yytext);
{WHILE}                 printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"WHILE",yytext);
{FOR}             printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"FOR",yytext);
{FOREACH}         printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"FOREACH",yytext);
{IN}              printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"IN",yytext);
```

```
{IF}                printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"IF",yytext);
{THEN}                    printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"THEN",yytext);
{ELSE}                    printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"ELSE",yytext);
{FUNCTION}          printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"FUNCTION",yytext);
{STRING}            printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"STRING",yytext);
{BIN_ARITH_OP}      printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"BIN_ARITH_OP",yytext);
{BIN_REL_OP}        printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"BIN_REL_OP",yytext);
{UN_ARITH_OP}       printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"UN_ARITH_OP",yytext);
{ASSIGN}            printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"ASSIGN",yytext);
{LOGIC_OP}          printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"LOGIC_OP",yytext);
{NOT}               printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"NOT",yytext);
{IDENTIFIER}        printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"IDENTIFIER",yytext);
{SEP}               printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"SEP",yytext);
{START_BLOCK}       printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"START_BLOCK",yytext);
{END_BLOCK}         printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"END_BLOCK",yytext);
{L_PARENT}          printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"L_PARENT",yytext);
{R_PARENT}          printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"R_PARENT",yytext);
{L_BRACKET}         printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"L_BRACKET",yytext);
{R_BRACKET}         printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"R_BRACKET",yytext);
{COMMA}                   printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"COMMA",yytext);
{ONE_LINE_COMMENT}              printf("Line %d: Found token %s
(lexeme: '%s').\n",yylineno,"COMMENT",yytext);
{COMMENT}           printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"COMMENT",yytext);
{LC}          printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"LEFT COMMENT",yytext);
{RC}          printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"RIGHT COMMENT",yytext);
{PAIR}                    printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"PAIR",yytext);
{COMPLEX}   printf("Line %d: Found token %s (lexeme:
'%s').\n",yylineno,"COMPLEX",yytext);
{SPACE}                   printf("");
.                   printf("Line %d: Invalid token
('%s').\n",yylineno,yytext);
%%

void main(int argc,char* argv[])
{
      yylex();
}
```

שאלה 2:

השפה הבאה mini-HTML מכילה את התגים (האסימונים) הבאים:

תג פותח	תג סוגר (אם ישנו כזה)	שם האסימון	הערה
'<html>'	'</html>'	HTML	1. הגדרת מסמך
'<head>'	'</head>'	HEAD	2. ראש מסמך
'<title>'	'</tiltle>'	TITLE	3. כותרת מסמך
'<body>'	'</body>'	BODY	4. גוף מסמך
'<table'	'</table>'	TABLE	5. הגדרת טבלה
'<tr>'	'</tr>'	ROW	6. הגדרת שורה בטבלה
'<tc>'	'</tc>'	COL	7. הגדרת עמודה בטבלה
'<a href='	''	ANCHOR	8. תגי עוגן
'<def'		DEF	9. הגדרת משתנה
'<set'		SET	10. השמה למשתנה
מחרוזת המתחילה ב $ ומכילה רק אותיות וספרות (פרט ל-$ עצמו)		ID	11. מזהה
':='		ASSIGN	12. השמה
'='		EQUAL	13. שווה
'>'		TAGEND	14. סוף תג מורכב
'<user_input>' מהמשתמש		USER_INPUT	15. קבלת קלט
'italics', 'bold', 'underlined' (font group) 'red', 'green', 'blue' (color group) 'small', 'medium', 'large', '+1', '-1' (size group)		MODIF	16. משנה
'font', 'color', 'size'		SELECT	17. בוחר קבוצה
URL ['http://'|'ftp//'][domain pathname][:portnum] portnum הוא מספר טבעי בין 0 ל – 16384 domain הינו שם אתר, המורכב בדיוק מ-4 תת-מחרוזות (המכילות ספרות ואותיות בלבד) ומופרדות ע"י 3 נקודות. pathname הינו שם קובץ הכולל path (אין רווח בין			18. כתובת

תג פותח	תג סוגר (אם ישנו כזה)	שם האסימון	הערה

(pathname-ו domain).

19. הערה COMMENT מחרוזת המתחילה ב- '- -! <' ומסתימת ב- '-> -' כאשר

א. אין חפיפת תווים (המחרוזת '<--!>' אינה הערה

ב. הערה יכולה להתפרש על מספר שורות

ג. אין הערות מקוננות

20. מלל TEXT כל רצף תווים שלא הוגדר עד כה, המתחיל באות גדולה,

אינו כולל רווחים לבנים ואינו כולל את התווים '>' ו- '<'

21. מרכאה QUOTE ' " '

הערות:

1. כל האלמנטים שאינם ממוספרים אינם אסימונים.
2. ניתן להניח שאורך כל מחרוזת המזהים בקלט קטן או שווה ל-32 תווים.
3. כל רצף של תווים המתחיל ב- '<', מסתיים ב- '<' , אינו חלק מהערה ואינו מהווה תג המוגדר יתפרש כשגיאה.

מטלות:

1. יש לכתוב מנתח לקסיקלי לשפה למעלה בעזרת Lex על פי הדרישות הבאות:

א. על המנתח להדפיס עבור כל אסימון (למעט הערות ומזהים) את הדברים הבאים:

1. מספר שורת הקלט בה האיסמון מופיע , כאשר שורות שמכילות הערות לא נספרות . (אפשר להניח שבשורת הערות אין אסימונים שיש לספור אותם).
2. האסימון שזוהה (תחת העמודה אסימון)
3. ה – lexeme שזוהה (תחת העמודה תג)
4. ערכים סמנטיים:

א. עבור התגים הפותחים והמסיימים יש לכתוב אם מדובר בתג פותח או מסיים.

ב. עבור האסימון MODIF יש להדפיס לאיזו קבוצה שייך ה- lexeme (דוגמא בהמשך).

ב. יש לנהל טבלת סמלים עבור המזהים עם מקום לחמישה מזהים בלבד . אם מזהים יותר מחמישה יש לעצור את הניתוח ולהודיע הודעת שגיאה מתאימה . אם מגלים מזהה יש להודיע האם הוא "קיים" או "חדש".

ג. הערות יודפסו כלשונן, כולל תגי ההערה.

ד. עבור תווים לא מזוהים יש להדפיס הודעת שגיאה הכוללת את מספר השורה ואת התו שגרם לבעייה. יש להמשיך את הניתוח מהתו הבא ואילך.

ה. על הקובץ להיות מתועד.

2. יש להוסיף פרוט על:

א. אופן הטיפול בשגיאות.

ב. אופן מימוש טבלת הסמלים.

ג. אם ישנם דרישות לא מוגדרות, יש לכתוב את ההנחות.

<u>דוגמת קלט:</u>

```
<! - - A sample input file - - for the lex exercise - - >
<html>

<head>
<title> Test Input File</title>
</head>

<body>
Hello World
<def color $cvar1>
<def color $cvar2>
<set $cvar1 := green>
<! - - This comment takes up
three input
Lines - - >

<table font = italics>
<tr>
<tc>
TextInput
</tc>
</tr>
</table>
<user_input>

<a href="http://www.braude.ac.il/public/files/filname:PDP11">
Some Link</a>
</body>

</html>
```

<u>והפלט המתאים:</u>

<!- - A sample input file - - for assignment 1 - - >

Line 1: HTML, lexeme: '<html>', begin tag.

Line 3: HEAD, lexeme: '<head>', begin tag.

Line 4: TITLE, lexeme: '<title>', begin tag.

Line 4: TEXT, lexeme: 'Test'

Line 4: TEXT, lexeme: 'Input'

Line 4: TEXT, lexeme: 'File

Line 4: TITLE, lexeme: '</title>', end tag.

Line 5: HEAD, lexeme: '</head>', end tag.

Line 7: BODY, lexeme: '<body>', begin tag.

Line 8: TEXT, lexeme: 'Hello'.

Line 8: TEXT, lexeme: 'World'.

Line 9: DEF, lexeme: '<def'.

Line 9: SELECT, lexeme: 'color'.

Line 9: ID: '$cvar1', new place 1 in symbol table.

Line 9: TAGEND, lexeme: '>'.

Line 10: DEF, lexeme: '<def'.

Line 10: SELECT, lexeme: 'color'.

Line 10: ID: '$cvar2', new place 2 in symbol table.

Line 10: TAGEND, lexeme: '>'.

Line 11: SET, lexeme: '<set.

Line 11: ID: '$cvar1', exist in place 1 in symbol table.

Line 11: ASSIGN, lexeme: ':='.

Line 11: MODIF, lexeme: 'green', color group.

Line 11: TAGEND, lexeme: '>'.

<! - - This comment takes up

three input

Lines - - >

Line 13: TABLE, lexeme: '<table', begin tag.

Line 13: SELECT, lexeme: 'font.

Line 13: EQUAL, lexeme: '='.

Line 13: MODIF, lexeme: 'italics, font group.

Line 13: TAGEND, lexeme: '>'.

Line 14: ROW, lexeme: '<tr>', begin tag.

Line 15: COL, lexeme: '<tc>', begin tag.

Line 16: TEXT, lexeme: 'TextInput'

Line 17: COL, lexeme: '</tc>', end tag.

Line 18: ROW, lexeme: '</tr>', end tag.

Line 19: TABLE, lexeme: '</table', end tag.

Line 20: USER_INPUT, lexeme: '<user_input>'

Line 22: ANCHOR, lexeme: '<ta href=', begin tag.

Line 22: QUOTE, lexeme: '"'.

Line 22: URL, lexeme: 'http://www.braude.ac.il/public/files/filname:PDP11'.

Line 22: QUOTE, lexeme: '"'.

Line 22: TAGEND, lexeme: '>'.

Line 23: TEXT, lexeme: 'Some'

Line 23: TEXT, lexeme: 'Link'.

Line 23: ANCHOR, lexeme: '', end tag.

Line 24: BODY, lexeme: '</body>', end tag.

Line 26: HTML, lexeme: '</html>', end tag.

פיתרון קובץ lex:

```
}%%{
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
char *identifiers[5];           // symbol table for ID's
int     idcount=0;                          // global counter for the ID's

/*
        function name: printFunc
        parm: char* str, int type
        mission: print a formated text to the standart output
        type explanation:       1 means a tag with begin tag and end tag.
                                        2 means a tag without end tag.
                                        3 means modif token of          font group
                                        4 means modif token of  color group
                                        5 means modif token of          size group
*/
void printFunc (char* str,int i);

/*function name: newLineCounter
```

```
  parm: char* str
  mission: return numbers of new lines in given comments token*/
int newLineCounter(char* str);

/*function name: idHandler
  mission: check if the ID is a new ID
  check if we don't have 5 ID's already
  enter a new ID into symbol table
  */
void idHandler();
%}

/* lex-compat: Tell lex to make a truck on line numbers in the extern varible yylineno*/
/* noyywrap: In order to compile the .c file in vs2005 we needed to add this option*/

%option lex-compat
%option noyywrap

/*Defenitaions*/
whitespace  ([\t\n ])
html            (<html>|<\/html>)
head            (<head>|<\/head>)
title           (<title>|<\/title>)
body            (<body>|<\/body>)
table           (<table>|<\/table>|<table)
row                     (<tr>|<\/tr>)
col                     (<tc>|<\/tc>)
anchor          (<a[ ]href=|<\/a>)
def                     (<def)
set                     (<set)
numdigit        ([a-zA-Z0-9])
id                      ($)({numdigit})+
assign          (:=)
equal           (=)
tagend          (>)
user_input      (<user_input>)
fontmodif       (italics|bold|underlined)
colormodif      (red|green|blue)
sizemodif       (small|medium|large|\+1|\-1)
select          (font|color|size)
domain          ({numdigit}+)[\.]({numdigit}+)[\.]({numdigit}+)[\.]({numdigit}+)
pathname        ((\/)({numdigit}+))
portnum                 (((([1])(([0-5])([0-9]){3}|(([6])((([0-2])([0-9]){2}|([3])(([0-7])([0-
9])|([8])([0-4]))))))))|((([1-9])([0-9])?([0-9])?([0-9])?))|([0]))
url                     (http:\/\/|ftp:\/\/)({domain}{pathname}+)(:)({portnum})
comment                 ("<!--"((.)?(([^>]|[^-][>]|[^-][-][>])\n)?)*"-->")
text            ([A-Z])([^\t\n<> ])*
quote           (\")
error           (.)
```

```
%%

{html}          printFunc("HTML",1);
{head}          printFunc("HEAD",1);
{title}         printFunc("TITLE",1);
{body}          printFunc("BODY",1);
{table}         printFunc("TABLE",1);
{row}           printFunc("ROW",1);
{col}           printFunc("COL",1);
{anchor}        printFunc("ANCHOR",1);
{def}           printFunc("DEF",2);
{set}           printFunc("SET",2);
{id}            idHandler();
{assign}        printFunc("ASSIGN",2);
{equal}         printFunc("EQUAL",2);
{tagend}        printFunc("TAGEND",2);
{user_input}    printFunc("USER_INPUT",2);
{fontmodif}     printFunc("MODIF",3);
{colormodif}    printFunc("MODIF",4);
{sizemodif}     printFunc("MODIF",5);
{select}        printFunc("SELECT",2);
{url}           printFunc("URL",2);
{quote}         printFunc("QUOTE",2);
{comment}       {
                        fprintf(stdout,"%s\n",yytext);
                        yylineno=yylineno-newLineCounter(yytext);
                }
{whitespace}    ;
{text}          printFunc("TEXT",2);
{error}         fprintf(stdout, "Line %d: Invalid Token ('%s')\n", yylineno,
yytext);        /*on error print to the standart error*/;

%%

/*visual studio 2005 don't make the main in automatic way so we write it here */

main()
{
        yylex();
}


int newLineCounter(char* str){
        int counter=1;
        int i;
        for (i=0;i<strlen(str);i++)
        {
                if (yytext[i]=='\n')
                        counter++;
```

```
	}
	return counter;
}

void idHandler()
{
	char new[32];
	int flag=1;//0 mean old 1 mean new
	int i;

	if (idcount>=5){
		fprintf(stdout, "Line %d: ID, lexeme: '%s' is over the limit of 5
ID's\n",yylineno,yytext);
		exit(0);
	}

	for (i=0;i<5;i++)
	{
		if (identifiers[i]!=NULL){
			if (strcmp(identifiers[i],yytext)==0){
				flag=0;
				break;
			}
		}
	}
	if (flag==1){
		identifiers[idcount]=malloc(yyleng*sizeof(char));
		if (identifiers[idcount]==NULL){
			fprintf(stdout, "memory allocation error");
			exit(1);
		}
		strcpy(identifiers[idcount],yytext);
		fprintf(stdout, "Line %d: ID, lexeme: '%s', new place %d in symbol
table.\n",yylineno,yytext,idcount);
	}else{
		fprintf(stdout, "Line %d: ID, lexeme: '%s', exist in place %d in symbol
table.\n",yylineno,yytext,i);
	}
	idcount++;
}

void printFunc (char* str,int type)
{
	char *end;
	end=malloc(sizeof(char));
	if (end==NULL){
		fprintf(stdout, "memory allocation error");
		exit(1);
	}
	strcpy(end,".");
```

```
switch (type){
case 1:
        if(yytext[1]=='/'){
                free(end);
                end=malloc(sizeof(char)*(strlen(", end tag.")));
                if (end==NULL){
                        fprintf(stdout, "memory allocation error");
                        exit(1);
                }
                strcpy(end,", end tag.");
        }
        else{
                free(end);
                end=malloc(sizeof(char)*(strlen(", begin tag.")));
                if (end==NULL){
                        fprintf(stdout, "memory allocation error");
                        exit(1);
                }
                strcpy(end,", begin tag.");
        }
        break;
case 2:
        break;
case 3:
        free(end);
                end=malloc(sizeof(char)*(strlen(", font group.")));
                if (end==NULL){
                        fprintf(stdout, "memory allocation error");
                        exit(1);
                }
        strcpy(end,", font group.");
        break;
case 4:
        free(end);
        end=malloc(sizeof(char)*(strlen(", color group.")));
        if (end==NULL){
                fprintf(stdout, "memory allocation error");
                exit(1);
        }
        strcpy(end,", color group.");
        break;
case 5:
        free(end);
        end=malloc(sizeof(char)*(strlen(", size group.")));
        if (end==NULL){
                fprintf(stdout, "memory allocation error");
                exit(1);
        }
        strcpy(end,", size group.");
        break;
```

```
        }
        fprintf(stdout, "Line %d: %s, lexeme: '%s'%s\n",yylineno,str,yytext,end);
        free(end);
}
```

הערות:

2.א.

טיפול בשגיאות התבצע בצורה כזו שכל תו שלא התקבל ע"י אחד החוקים שהגדרנו התקבל כשגיאה.(הוספנו חוק של שגיאה שקיבל כל תו במידה ולא התקבל עד כה. הטיפול בו היה הדפסת שורת שגיאה לקובץ הפלט.

2.ב:

טבלת הסמלים מומשה כמערך מחרוזות שהוקצו דינמית תוך כדי ריצת המנתח.
כאשר נתקלנו במזהה ID ביצענו בדיקה האם הוא קיים במערך זה ע"י מעבר על כל המערך (זאת מפני שידענו מה הגודל וזה גודל קטן אם היה לנו צורך בניהול של מערכת יותר גדולה כנראה שהמ בנה היה שונה לצורך גישה מהירה יותר) עבור כל תא במערך ביצענו השוואה מול המזהה שאנו רוצים להכניס במידה והיה קיים הודענו על כך שהוא קיים במידה ולא היה קיים ובנוסף טבלת הסמלים עדיין לא היתה מלאה (כלומר עוד לא הוקצו 5 סמלים) אנו מודיעים על מזהה חדש,במידה והוקצו חמישה מזהים והמזהה שאנו בודקים לא תאם אף אחד מהם אנו מודיעים על שגיאה מתאימה (ההודעה כשגיאה בקובץ הפלט).

2.ג:

הנחות:

- מכיוון שלא ראינו צורך להפריד בין תגי התחלה ותגי סיום בחוקים הגדרנו אותם באותו החוק ושינינו את מה שמודפס רק בפונקצית ההדפסה שמטפלת בחוק זה(בדיקה האם התו השני במחרוזת הוא '/'), זאת במקום להגדיר חוקים עבור תגי פתיחה ותגי סגירה.

- עבור URL הדפסת השגיאה אינה זהה לשאר השגיאות מפני שאנו מבצעים את הבדיקה בקוד C שמטפל בזהוי חוק שהוגדר כבר ולכן לא יכולנו להדפיס שגיאה עבור כל תו אז הדפסנו שגיאה עבור כל המחרוזת שזוהתה כ-URL תוך ציון שזהו URL לא חוקי.

- נכתבה פונקצית הדפסה אחת שמבצעת הדפסה למסך ולא לקובץ על מנת להקל עלינו את הכתיבה לקובץ (לא בכך צורך) ולכן צירפנו קובץ הרצה של הפקודה שמריצה את הקובץ שנוצר לאחר הקימפול עם ה-INPUT שלנו ולקובץ OUTPUT שלנו גם כן שייותר באותה תיקיה של ההרצה. פונקציה זו מבצעת את כל ההדפסות הנדרשות ע"י פרמטר שנשלח אליה כדי לברור באיזה סוג חוקים מדובר.

שאלה 3:

הניחו שהשפות הבאות הן מעל הא"ב $\Sigma=\{a,b\}$

- L1 : כל המחרוזות המכילות לפחות שני a-ים.
- L2 : כל המחרוזות המכילות לפחות b אחד.
- L3 : כל המחרוזות המכילות לפחות שני a-ים ולפחות b אחד.
- L4 : כל המחרוזות המכילות לכל היותר a אחד ואינן מכילות b-ים.

יש לבנות DFA עבור השפות שלמעלה.

שאלה 4:

א. יש לקבוע עבור כל אחת מהשפות הבאות מי מהן רגולרית – יש להסביר בשני משפטים.
- L1 : כל המחרוזות מעל הא"ב {s,t} המכילות מספר שווה של s-ים ושל t-ים.
- L2 : כל המחרוזות מעל הא"ב {0,1} שהן פולינדרום (ניתן לקרוא אותן מימין ומשמאל אותו הדבר).
- L3 : כל המילים במילון אבן שושן.

ב. שפת c אינה שפה רגולרית.
- מהו הא"ב של השפה?
- מהי מילה בשפה?
- הסבר מדוע השפה אינה רגולרית – בשניים שלושה משפטים.

שאלה 5:

נתונה הגדרת האסימונים למנתח הלקסיקלי שנבנה בעזרת המחולל Lex :

```
c(c+)(c|d)        { printf("1"); }
c(c*)(c|d)        { printf("2"); }
ccc               { printf("3"); }
c(c+)(c|d)(c+)    { printf("4"); }
```

א. עבור כל אחת מההגדרות הבאות, קבעו האם המנתחים המתקבלים משתי ההגדרות ייצרו את אותו ניתוח לקסיקלי על כל קלט. אם כן, הסבירו מדוע. אם לא, הראו את המחרוזת עליה מתקבלות תוצאות שונות.

הגדרה א: (השורה השנייה הושמטה)

```
c(c+)(c|d)        { printf("1"); }
ccc               { printf("3"); }
c(c+)(c|d)(c+)    { printf("4"); }
```

הגדרה ב: (השורה השלישית הושמטה)

```
c(c+)(c|d)        { printf("1"); }
c(c*)(c|d)        { printf("2"); }
c(c+)(c|d)(c+)    { printf("4"); }
```

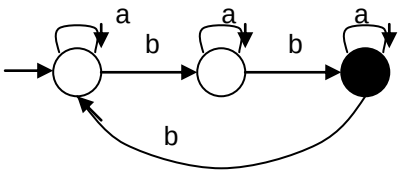
ב. הראו את הניתוח הלקסיקלי של המחרוזות הבאות ע"פ ההגדרות המקוריות:

1. ccc
2. ccccdcd

ניתן להניח שהמנתח הלקסיקלי מטפל בשגיאות כפי שנלמד בכיתה .

שאלה 6:

נתון ה DFA הבא מעל הא"ב $\Sigma=\{a,b\}$:



יש לתת הגדרה של השפה במשפט אחד.
יש לרשום ביטוי רגולרי עבור השפה.

שאלה 7:

נתון הקטע הבא מתוך קובץ הקלט למנתח הלקסיקלי:

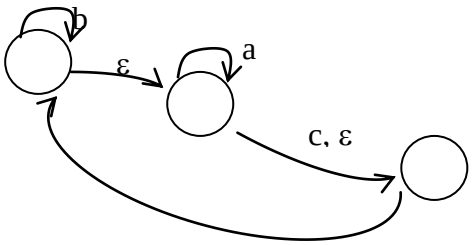
```
%%
ab              printf("1 %s\n", yytext);
b*a*c ?         printf("2 %s\n", yytext);
ba . a * printf("3 %s\n", yytext);
aa*c            printf("4 %s\n", yytext);
ab*c ?          printf("5 %s\n", yytext);
%%
```

להזכירכם yytext הינו החלק בקלט שהתאים לביטוי הרגולרי.

א. מה יהיה הפלט עבור הקלט הבא? (נמקו!! לא תתקבל תשובה ללא הסבר)

a b b a c b a b a b c a b
5 a b b
2 a c
3 b a b a
2 b c
1 a b

ב. נתון שהשפה למעלה היא מעל הא"ב $\Sigma = \{a, b, c\}$. האם ישנם מחרוזות מעל הא"ב הנ"ל, שהחוקים למעלה לא יתאימו להם? אם כן, הראו אילו, אם לא הסבירו.



אפשר כאן כל מחרוזת.

ג. האם ניתן להוריד את אחד החוקים למעלה מבלי לשנות את ההתנהגות של המנתח הלקסיקלי? אם כן, הראו אילו חוקים ניתן להוריד. אם לא הסבירו מדוע.

מנתח תחבירי

שאלה 1:

נתון הדקדוק הבא: Σ = { **LP**, **RP**, **+**, *****, **num**, **id** }

1. S → E
2. E → **LP** E **RP**
3. E → **LP** E *A* E **RP**
4. E → E ***** Q
5. E → Q
6. *A* → **+**
7. Q → **num**
8. Q → **id**

1. חשבו את פונקצית ה first עבור כל משתנה דקדוק.
2. חשבו את פונקצית ה follow עבור כל משתנה דקדוק.

1. **Fixed parser:**
2. S -> E
3. E -> **LP** E **RP** Z | **LP** E *A* E **RP** Z
4. E -> Q Z
5. Z -> ***** Q Z | <>
6. *A* -> **+**
7. Q -> **num** | **id**

	FIRST	FOLLOW
S	{LP , num , id}	$
E	{LP , num , id}	{RP , + , $}
Z	{* , <>}	{RP , + , $}
Q	{num , id}	{* , RP , + , $}
A	{+}	{LP , num , id }

3. בנו את הטבלה.

	LP	RP	+	*	num	id	$
S	S:E				S:E	S:E	
E	E:**LP** E **RP** Z E:**LP** E **RP** Z				E:Q Z	E:Q Z	
Z		Z:<>	Z:<>	Z:***** Q Z			Z:<>
A				*A*:+			
Q					Q:**num**	Q:**id**	

4. האם הדקדוק הוא LL(1).

אם כן, נמקו.

אם לא, הראו מדוע (כל הסיבות). שנו את הדקדוק כך שיהיה LL(1), וחיזרו על סעיפים א-ג.

הדקדוק הוא לא LL1 מכיוון ש:

- בדקדוק המקורי היתה רקורסיה שמאלית
- בתא בטבלה (המסומן באדום) מופיעים 2 חוקי גזירה, כלומר קיימים 2 חוקי גזירה עבור נקודה מסויימת בגזירת המילה.

הדקדוק המתוקן הוא

LL(1) ::

S -> E $
E -> LP E E1 | Q Z
E1 -> A E RP Z | RP Z
Z -> * Q Z | <>
A -> +
Q -> num | id

	FIRST	FOLLOW
S	{LP , num , id}	$
E	{LP , num , id}	{RP , + , $}
E1	{+ , RP}	{RP , + , $}
Z	{* , <>}	{RP , + , $}
Q	{num , id}	{* , RP , + , $}
A	{+}	{LP , num , id }

	LP	RP	+	*	num	id	$
S	S:E				S:E	S:E	
E	E:LP E E1				E:Q Z	E:Q Z	
E1		E1:RP Z	E1:A E RP Z				
Z		Z:<>	Z:<>	Z:* Q Z			Z:<>
A				A:+			
Q					Q:num	Q:id	

5. בהינתן הקלט : **(a + 3)** המנתח הלקסיקלי ייצר את הפלט: **LP id + num RP** .
מלאו את טבלת ההרצה של המנתח התחבירי כפי שעשינו בכיתה על המילה **LP id + num RP**

Stack	**Input**	**Comment**
S	LP id + num RP $	SHIFT S:E $
$ E	LP id + num RP $	SHIFT E:LP E E1
$ E1 E LP	LP id + num RP $	REDUCE LP
$ E1 E	id + num RP $	SHIFT E:Q Z
$ E1 Z Q	id + num RP $	SHIFT Q:id
$ E1 Z id	id + num RP $	REDUCE id
$ E1 Z	+ num RP $	SHIFT Z:<>
$ E1	+ num RP $	SHIFT E1:A E RP Z
$ Z RP E A	+ num RP $	SHIFT A:+
$ Z RP E +	+ num RP $	REDUCE +
$ Z RP E	num RP $	SHIFT E:Q Z
$ Z RP Z Q	num RP $	SHIFT Q:num
$ Z RP Z num	num RP $	REDUCE num
$ Z RP Z	RP $	SHIFT Z:<>
$ Z RP	RP $	REDUCE RP
$ Z	$	SHIFT Z:<>
$	$	**ACCEPT**

שאלה 2

האם הדקדוקים הבא הם LL(1) :

א.

1. $S \rightarrow ab$
2. $S \rightarrow Ab$
3. $A \rightarrow BcA$
4. $A \rightarrow bS$
5. $B \rightarrow aB$
6. $B \rightarrow \varepsilon$

	FIRST	FOLLOW
S	{a , b , c , <>}	{b , $}
A	{a , b , c , <>}	{b}
B	{a , <>}	{c}

Parse Table:

	a	b	c	$
S	S:a b S:A b	S:A b	S:A b	S:A b
A	A:B c A	A:b S A:B c A		
B	B:a B		B:<>	

We have 2 conflicts . That is why this is NOT LL(1) parser.
The conflicts are marked with red color.

ב.

1. $S \rightarrow AaAb$
2. $S \rightarrow BbBa$
3. $A \rightarrow d$
4. $A \rightarrow \varepsilon$
5. $B \rightarrow c$

$B \rightarrow \varepsilon$

	FIRST	**FOLLOW**
S	{d , c , <>}	{$}
A	{d , <>}	{a , b}
B	{c , <>}	{a , b}

Parse Table:

	a	b	c	d	$
S			S:B b B a	S:A a A b	**S:A a** **A b** **S:B b** **B a**
A	A:<>	A:<>		A:d	
B	B:<>	B:<>	B:c		

We have 1 conflict . That is why this is **NOT LL(1) parser.**

<u>שאלה 3</u>

א. תנו דוגמא לדקדוק ח"ה אשר מקיים את הדברים הבאים:

- יש בו לפחות שתי מלים.
- G אינו דקדוק LL(1).
- G הינו דקדוק LL(k) כאשר k>1.

השתדלו לבחור מספרי משתני דקדוק, טרמינלים וכללי גזירה מינימליים אשר יקיימו את הדרישות לעיל .

הסבירו מדוע הדקדוק מקיים את הדרישות.

ב. האם ניתן למצוא דקדוק שיש בו מילה אחת בלבד וכל שאר התנאים ב- א. ויכול להיות דקדוק LL(k) כאשר k>1? נמקו.

<u>הדקדוק הוא:</u>

S-> a | aa

	a	$
S	S:a S:aa	

We have LL(2) parser with 2 words (a & aa).

ב. לא ניתן למצוא דקדוק כזה מכיוון שלמילה אחת יש גזירה יחידה ולכן לא נגיע למצב שיש יותר מחוק גזירה אחת עבור אותה המילה.
ההוכחה לכך היא:
נגדיר αi – ביטוי שהוא non terminal
Ti – ביטוי שהוא terminal

S: α1α2α3α4…αn
α1:t1
α2:t2
.
.
.
αn:tn

לאחר הגזירה נקבל S:t1t2…tn, זו מילה יחידה שנגזרה ע"י חוקי גזירה יחידים וניתן לראות שבשום מקום בדרך לקבלת המילה הסופית לא היתה "התפצלות דרכים", כלומר לא היתה אפשרות לגזור את המילה בדרך אחרת. ניתן להגיד שהגזירה היא חח"ע וכאשר מילה נגזרת באופן חח"ע מדקדוק, אנו נקבל עבורה חוק גזירה יחיד. מש"ל

<u>הערה:</u>
אם הדקדוק

S: Aa|aa
A: a

נחשב לחוקי, אז אכן קיימת אפשרות ליצור דקדוק LL(2) שמכיל מילה אחת בלבד.

<u>שאלה 4:</u>

נתון הדקדוק הבא:

1. S → A B
2. S → C <u>e</u>
3. S → <u>d</u>
4. S → ε
5. A → <u>a</u> A
6. A → <u>d</u>
7. B → B <u>b</u>
8. B → <u>f</u>
9. C → <u>c</u> C
10. C → ε

א. מהי השפה הנגזרת ע"י הדקדוק הנ"ל ?

ב. האם הדקדוק חד משמעי ?

ג. חשבו את הפונקציה first עבור כל המשתנים.

ד. חשבו את הפונקציה follow עבור כל המשתנים.

ה. האם הדקדוק הוא LL(1) ? אם כן נמקו. אם לא, הפכו את הדקדוק ל LL(1).

ו. מלאו את טבלת הניתוח עבור דקדוק ה LL(1) (אם הוא כזה, או אם לא, אחרי שהפכתם אותו לכזה)

ז. השלימו את טבלת ההרצה של המנתח עבור הקלט 'adfb'.

שאלה 5:

נתון הדקדוק הבא:

$S \rightarrow A$
$A \rightarrow A\ B\ -\ |\ +\ B$
$B \rightarrow b\ |\ B\ b\ |\ \varepsilon$

א. האם זהו דקדוק LR(0) ? (בנו את האוטומט, והראו בעזרתו)

S1: $S \rightarrow \bullet\ A$	on A goto s2
$A \rightarrow \bullet\ A\ B\ -$	on A goto S2
$A \rightarrow \bullet\ +\ B$	on + shift S3
S2: $S \rightarrow A\ \bullet$	accept
$A \rightarrow\ A \bullet B\ -$	on B goto S4
$B \rightarrow \bullet\ b$	on b shift S5
$B \rightarrow \bullet\ B\ b$	on B goto S4
$B \rightarrow \bullet\ \varepsilon$	reduce
S3: $A \rightarrow +\ \bullet\ B$	on B goto S4
$B \rightarrow \bullet\ b$	on b shift S5
$B \rightarrow \bullet\ B\ b$	on B goto S4
$B \rightarrow \bullet\ \varepsilon$	reduce
S4: $A \rightarrow\ A\ B \bullet -$	on - shift S6
$B \rightarrow\ B \bullet b$	on b shift s7
$A \rightarrow +\ B \bullet$	reduce
S5: $B \rightarrow\ b \bullet$	reduce
S6: $A \rightarrow\ A\ B\ - \bullet$	reduce
S7: $B \rightarrow\ B\ b\ \bullet$	reduce

כבר רואים שיש shift/reduce conflict.

ב. האם זהו דקדוק SLR(1) ? (בנו את הטבלה – והראו את כל הקונפליקטים אם ישנם כאילו)

Non-Terminal	first	follow
S	+	$
A	+	$, b , -
B	b, ε	-, b, $

#	b	+	-	$	S	A	B
1		s3				2	
2	S5 r0, B		r0, B	acc r0, B			4
3	S5 r0, B		r0, B	r0, B			4
4	S7 r2,A		S6 r2,A	r2,A			
5	r1,B		r1,B	r1,B			
6	r3,A		r3,A	r3,A			
7	r2,B		r2,B	r2,B			

ג. האם זהו דקדוק LR(1) ? (בנו את האוטומט והראו את הנימוק)

S1:	$S \to \bullet A$	$	on A goto s2
	$A \to \bullet A\ B -$	$,b,-	on A goto S2
	$A \to \bullet + B$	$,b,-	on + shift S3
S2:	$S \to A \bullet$	$	accept
	$A \to A \bullet B -$	$,b,-	on B goto S4
	$B \to \bullet b$	-,b	on b shift S5
	$B \to \bullet B\ b$	-,b	on B goto S4
	$B \to \bullet \varepsilon$	-,b	reduce
S3:	$A \to + \bullet B$	$,b,-	on B goto S6
	$B \to \bullet b$	$,b,-	on b shift S7
	$B \to \bullet B\ b$	$,b,-	on B goto S6
	$B \to \bullet \varepsilon$	$,b,-	reduce
S4:	$A \to A\ B \bullet -$	$,b,-	on - shift S8
	$B \to B \bullet b$	-,b	on b shift s9
S5:	$B \to b \bullet$	-,b	reduce
S6:	$A \to + B \bullet$	$,b,-	reduce
	$B \to B \bullet b$	$,b,-	on b shift s10
S7:	$B \to b \bullet$	$,b,-	reduce
S8:	$A \to A\ B - \bullet$	$,b,-	reduce
S9:	$B \to B\ b \bullet$	-,b	reduce
S10:	$B \to B\ b \bullet$	$,b,-	reduce

#	b	+	-	$	S	A	B
1		s3				2	
2	S5 r0, B		r0, B	acc			4
3	S7 r0,B		r0,B	R0,B			6
4	…						
5							
6							
7							
8							
9							
10							
11							

שאלה 6:

נתון הדקדוק הבא:

$S \rightarrow Aa \mid bAc \mid dc \mid bda$
$A \rightarrow d$

א. האם זהו דקדוק LALR(1) ? כן או לא, נמקו

S1:	S -> •Aa	$, S2
	S -> •bAc	$, S3
	S -> •dc	$, S4
	S -> •bda	$, S3
	A -> •d	a, S4
S2:	S -> A•a	$, S5
S3:	S -> b•Ac	$, S6
	S -> b•da	$, S7
	A -> •d	c, S7
S4:	S -> d•c	$, S8
	A -> d•	a, reduce
S5:	S -> Aa•	$, reduce
S6:	S -> bA•c	$, S9
S7:	S -> bd•a	$, S10
	A -> d•	c, reduce
S8:	S -> dc•	$, reduce
S9:	S -> bAc•	$, reduce
S10:	S -> bda•	$, reduce

כדי להפוך ל – LALR(1) לא צריך לאחד כלום, כיוון שאין קונפליקט, התשובה חיובית.

ב. האם זהו דקדוק SLR(1) ? כן או לא, נמקו

ה- Follow(A) = {a, c} לכן יהיה קונפליקט בין S4 ל- S7 ולכן התשובה שלילית.

שאלה 7:

נתון הדקדוק הבא: $\Sigma = \{ a, b, +, * \}$

$S \rightarrow A + B \mid S + B$
$A \rightarrow S \mid B\ A \mid a$
$B \rightarrow B * A \mid b$

א. האם זהו דקדוק SLR(0) ? (בנו את האוטומט, והראו בעזרתו)

S0:	$S \rightarrow \bullet A + B$	on A goto S1
	$S \rightarrow \bullet S + B$	on S goto S2
	$A \rightarrow \bullet S$	on S goto S2
	$A \rightarrow \bullet B\ A$	on B goto S3
	$A \rightarrow \bullet a$	on a shift S4
	$B \rightarrow \bullet B * A$	on B goto S3
	$B \rightarrow \bullet b$	on b shift S5

S1: S → A • + B on + shift S6

S2: S → S • + B on + shift S7
A → S • reduce 1, A

S3: A → B • A on A goto S8
B → B • * A on * shift S9
A → • S on S goto S2
A → • B A on B goto S3
A → • aon a shift S4
B → • B * A on B goto S3
B → • bon b shift S5
S → • A + B on A goto S1
S → • S + B on S goto S2

S4: A → a •reduce 1, A

S5: B → b •reduce 1, B

S6: S → A + • B on B goto S10
B → • B * A on B goto S3
B → • bon b shift S5

S7: S → S + • B on B goto S11
B → • B * A on B goto S3
B → • bon b shift S5

S8: A → B A • reduce 2, A

S9: B → B * • A on A goto S12
A → • S on S goto S2
A → • B A on B goto S3
A → • aon a shift S4
B → • B * A on B goto S3
B → • bon b shift S5
S → • A + B on A goto S1
S → • S + B on S goto S2

S10: S → A + B • reduce 3, S

S11: S → S + B • reduce 3, S

S12: B → B * A • reduce 3, B

א. האם זהו דקדוק SLR(1) ? (בנו את הטבלה – והראו את כל הקונפליקטים אם ישנם כאילו)

Non-Terminal	first	follow
S	a b	$ + a b *
A	a b	a b + *
B	b	a b * $

#	a	b	+	*	$	S	A	B
0	S4	S5				2	1	3
1			S6					
2		r1,A	r1,A S7	r1,A				
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								

לא, ב S2 יש קונפליקט והוא לא נפתר ב – SLR.

<u>שאלה 8:</u>
נתון הדקדוק הבא:

$S \rightarrow AaAb \mid BbBa$
$A \rightarrow \varepsilon$
$B \rightarrow \varepsilon$

א. האם זהו דקדוק LL(1) ? כן

כי כאשר מנסים לקבוע את ה-first וה- follow רואים ש:

$first(AaAb) \cap first(BbBa) = \{a\} \cap \{b\} = \varnothing$

ב. האם זהו דקדוק SLR(1) ? לא

כי, כאשר בודקים את ה – follow של החוקים מקבלים ש:
$follow(A) \cap follow(B) = \{a,b\} \cap \{a,b\} \neq \varnothing$
ולכן יש בעיית reduce/ reduce .

$S \rightarrow \bullet\ AaAb$	
$S \rightarrow \bullet\ BbBa$	
$A \rightarrow \bullet$	reduce
$B \rightarrow \bullet$	reduce

ג. האם זהו דקדוק LR(1) ? כן

כי אם דקדוק הוא LL(1) אזי בהכרח הוא גם LR(1).
וגם כי עבור החוק מקודם זה יראה כך:

$S \rightarrow \bullet\ AaAb$	$
$S \rightarrow \bullet\ BbBa$	$
$A \rightarrow \bullet$	a
$B \rightarrow \bullet$	b

ייצור קוד pmachine

שאלה 1:

מוסיפים לשפת מיני לולאת for – כיצד תמומש ב – P-machine. תארו את ה – code . הביטויים ב for מוגדרים באופן הבא:

I_1 is an ASSIGN statement, I_2 is a BOOLEAN expression, I_3 is an ASSIGN statement.

for(I_1 ; I_2 ; I_3)
S;

א. יש להסביר אילו שינויים אם בכלל נדרשים בכל אחד משלבי הקומפיילר :

מנתח לקסיקלי

_____

מנתח סינטקטי

_____

מנתח סמנטי

_____

ב. רשמו את הדקדוק של המבנה הנ"ל.

ג. יש לתת הסבר כללי על השינויים בקוד עבור המבנה לעיל .

ד. רשמו את הקוד עבור ה- pmachine באופן יעיל

code (for(I_1 ; I_2 ; I_3) S;) ρ =

פיתרון:

א)

- מנתח לקסיקלי: מוסיפים TOKEN מסוג ROF שיסמל לנו את סוף פקודת ה FOR (בדומה לפקודות IF ו FI בשפת miny).
- מנתח סינטקטי: מוסיפים את הדקדוק המתאים ובנוסף יהיה טיפול בהוספת צומת לעץ מסוג FOR.
- מנתח סמנטי: יבצע בדיקה שפקודת ה for(I1 ; I2 ; I3) היא תקינה כאשר יתקיימו האקסיומות הבאות:

  I1 is an ASSIGN statement, I2 is a BOOLEAN expression, I3 is an ASSIGN statement.

ב)

s→ for(I1; I2;I3)statment
I1→assign
I2→boolean
I3→assign

ג)

```
For _state:  FOR for_exp stat_seq ROF {$$=makenode(FOR,$2,$3,NULL,0,NULL);}
```

כאשר נציין את הפרמטרים כך:

$2 יהיה for_exp
$3 יהיה stat_seq

```
for_exp:     '(' assign ';' expr ';' assign ')' {$$=makenode(FOREXP,$2,$4,$6,0,NULL);}
```

כאשר נציין את הפרמטרים כך:

$2 יהיה assign
$4יהיה expr
$6 יהיה assign

השינויים שנבצע בקוד יהיו:

כאשר נזהה פקודת FOR, ניצור צומת בעץ, בעל 2 צמתים, כאשר הצומת השמאלי $2 יהיה for_exp ,
והצומת הימני $3 יהיה stat_seq
עבור הצומת השמאלי ה-"for_exp " יבנה צומת בעלת שלושה בנים כאשר $2 יהיה assign, $4יהיה
expr, $6 יהיה assign.

ד)

code (for(I1 ; I2 ; I3) S;) =
code (I1); L1: code_R(I2); fjp L2; code S; code (I3); ujp L1; L2:
כאשר מגיעים לlable L2 יתבצע תנאי יציאה מלולאת for.

<u>שאלה 2:</u>
יש להפעיל את ה-code שנוצר על קטע הקוד הבא (כולל כתובות המשתנים) ולפרט את השלבים.
נתון עוד כי $\rho(x) = 5$ ו - $\rho(y) = 6$.

```
var  b : array[ 5 . . x,  -3 . . y] of integer;
     i, j  :  integer;

for ( i = 5 ; i <= x ; i = i+ 1)
        for ( j = -3; j < y; j = j + 1)
                b[i, j] = i + j;
```

<u>פיתרון:</u>

```
ldc a 15;
ldc i 5;
sto;
L1: ldc a 15;
ind i;
ldc a 5;
ind i;
leq i;
fjp L2;
ldc a 16;
ldc i 3;
neg i;
sto;
L3: ldc a 16;
ind i;
ldc a 6;
ind i;
les i;
fjp L4
dpl i;
ind i;
idc i 0;
ldc a 15;
ind i;
add i;
ldd 7;
mul i;
ldc a 16;
ind i;
```

```
add i;
ixa 1;
sli a;
ldc a 15;
ind i;
ldc a 16;
ind i;
add i;
sto
ldc a 16;
ldc a 16;
ind i;
ldc i 1;
add i;
sto;
ujp L3;
L4: ldc a 15;
ldc a 15;
ind i;
ldc i 1;
add i;
sto;
ujp L1;
L2;
```

<u>שאלה 3:</u>

רוצים להוסיף את הפקודה הבאה לשפת ה- miny

On E repeat S times B holds

כאשר B גוזר ביטוי בוליאני ו- E גוזר ביטוי מטיפוס integer.

משמעות הפקודה:
ביצוע S עד אשר התנאי הנגזר מ-B מתקיים מספר פעמים, כערך הביטוי הנגזר מ-E. על פי התנאים הבאים:

1. E מחושב פעם אחת לפני האיטראציה הראשונה.
2. B מחושב בכל איטראציה אחרי ביצוע S.
3. אם E מחושב להיות שלילי, S ו- B לא יחושבו כלל. אחרת גוף הלולאה מתבצע לפחות פעם אחת.

א. יש להסביר <u>בשורה או שתיים</u> (יותר לא יתקבל!) אילו שינויים אם בכלל נדרשים בכל אחד משלבי הקומפיילר:

מנתח לקסיקלי להוסיף את המפתחות החסרים – on, times, holds השאר קיים.

מנתח סינטקטי להוסיף את הדקדוק של הפעולה הזו (משתני הדקדוק E ו – B כבר מוגדרים בדקדוק)

S -> <u>on</u> E <u>repeat</u> S <u>times</u> B <u>holds</u>

מנתח סמנטי יש לבדוק אם E מחזיר מספר שלם, B הוא ביטוי המחזיר ערך בוליאני.

ב. יש לתת הסבר כללי על השינויים בקוד עבור המבנה לעיל.

(אם יש צורך במשתנים נוספים איך לקבוע אותם, ועל הרעיון על פיו אתם תצרו את הקוד)

צריך ליצור משתנה שיקרא את הערך שלם שמחזיר E (נקרא לו _v) ובכל איטראציה בה מתקיים B יש לחסר ממנו 1 ולסיים כאשר הגענו ל – 0.
היתר נראה דומה מאוד ללולאה רגילה.
המשתנה יוקצה בסוף החלק הסטאטי של המחסנית, כך שלא ישנה את הכתובות של המשתנים הפנימיים של הפונקציה,
הכתובת תהיה: $addr(\_v) = 5 + \sum_{i=1}^{k} size(v_i)$
כאשר k הוא מספר המשתנים המוגדרים בפונקציה בה כתוב המבנה.

ג. רשמו את הקוד עבור ה- pmachine באופן <u>יעיל</u> (על קוד לא יעיל לא ינתנו נקודות).

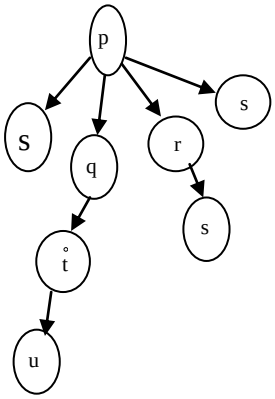
```
code (On  E repeat S times B holds ) ρ, nd =
     ldc a 0 addr(_v);
     codeR( E);
     sto i;
     ldc a 0 addr(_v);
     ind i;
     ldc i 0;
     geq  i;
     fjp  L2;   // if _v < 0, end loop
L1:  code(S);
     codeR( B);
     fjp L1;   // if B = FALSE don't sub 1 from the iteration counter.
     ldc a 0 addr(_v);
    ind i;
```

```
    ldc i 1;                    subtract 1 from the iteration counter
    sub i ;
    sto i;
     ldc a 0 addr(_v);
    ind i;
    ldc i 0;
    leq i;      // if _v <= 0 then exit the loop
    fjp L1;
L2:
```

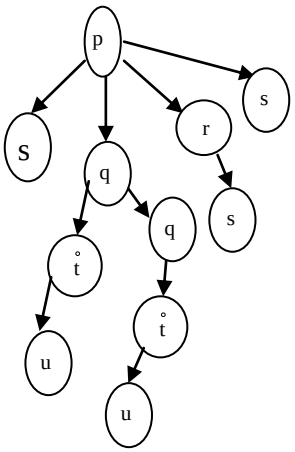
<u>שאלות 4 ו – 5 (שאלות 12 ו-16 מפרק 2 בספר)</u>

<u>שאלה 12:</u>

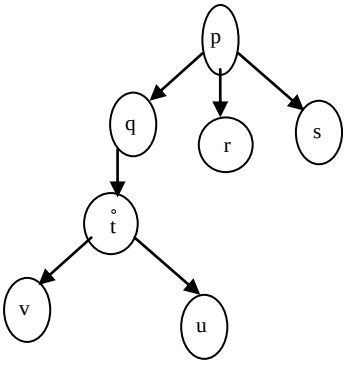
. b=false



b=true בקריאה הראשונה ל-q ובקריאה השנייה false.



ב.



שאלה 16:

a.

Instr.	Meaning	Cond.	Res.
Stoz T	STORE[SP-STORE[1]]=STORE[SP] STORE[SP-1]=STORE[SP] SP=SP-1		

b.ניתן להחזיר גם מבנים וכדומה, לכן נעתיק את מה שיש לנו באותה הפונקציה לפונקציה שקראה לה. נקרא לפקודה RETS שתקבל p –מספר הרשומות(שורות) שצריך להחזיר.
בהנחה שבשורת החזרה שנמצאת במחסנית זמן הריצה יושבת הכתובת שממנה יושבים הנתונים.

Instr.	Meaning	Comment

RETS p	SP := MP;
	PC := STORE[MP+4];
	EP :=STORE[MP+3];
	if(EP >= NP)
	then error("store overflow")
	fi
	MP := STORE[MP+2];
	for(i=0; i<p; i--) do
	STORE[SP+i+1] :=
	STORE[STORE[SP]+i];
	Od
	SP := SP+p;
		Result is in ton top of the return address and she also address.

c.
מכיוון שאנחנו צריכים את הערך של המשתנה והוא ניתן לשינוי, נחזיק אותו בחלק הדינאמי,
ונוסיף מבנה נתונים שיכיל את המשתנים הסטטיים הללו ואת שיוכם לאיזה פרוצדורה וכתובתם בחלק הדינאמי, כך שנוכל
תמיד להוציא את ערכם.

שאלה 6:

עליכם להריץ את התכנית הבאה ע"פ הדרישות מתחת:

רשימת הקבצים :

FLEX.EXE - הכלי LEX .

BISON.EXE - הכלי YACC.

Bison.simple - קובץ בשימוש BISON.EXE.

Miny.l - קובץ הגדרות עבור ה - LEX.

Miny.y - קובץ הגדרות עבור ה – YACC.

Typedef.h – הגדרת מבנה צומת בעץ (NODE)

Main.c - תוכנית C זמנית , עד אשר תהיה לכם תוכנית main משלכם (שאתם תכתבו).

תוכנית זו היא רק לבדיקה ראשונית שכל הפעולות שתבצעו בהמשך הצליחו וכן כלי ראשוני לבדיקת התוכנית שאתם מזינים לפני כתיבת הקוד שלכם. תוכנית זו מפעילה למעשה את המנתח הלקסיקלי ואחריו את המנתח התחבירי על התוכנית בשפת miny שתרצו לקמפל (קריאה לפונקציה yyparse()) , ומחזירה תשובה – הדפסה למסך , האם התוכנית שהכנסתם עברה קומפילציה (בדיקת syntax) – הודעת הצלחה, או נכשלה – הודעת שגיאה – (parsing error) עם מספר השורה בה יש בעיה (מס' שורה בקובץ Input).

תוכנית זו גם תראה לכם דוגמא , איך לקבל מצביע לשורש העץ שנבנה בזמן הרצת המנתח התחבירי וכן תדפיס לקובץ בשם tree.txt את העץ שנבנה מתוכנית

ה- miny שתעבירו כפרמטר ל - main. קובץ זה הוא קובץ טקסט כך שבסיום הרצת ה- main תוכלו להסתכל בו וכך להכיר את מבנה העץ שאיתו תעבדו.

סדר פעולות :

1. בניית המנתח הלקסיקלי בעזרת הכלי LEX ,ע"י הרצת הפקודה משורת ה- DOS:

   i. c:\ > FLEX miny.l

   פקודה זו יוצרת תוכנית בשם lex.yy.c

2. בניית המנתח התחבירי בעזרת הכלי BISON ,ע"י הרצת הפקודה משורת ה- DOS:

   1. c:\ > BISON -d miny.y

   פקודה זו יוצרת תוכנית בשם miny_ tab.c *וכמו כן קובץ בשם* miny _tab.h

3. קימפול תוכניות ה – C הנתונות (main.c) וכן את תוכניות ה-C שנוצרו (lex.yy.c,miny_tab.c) .

4. link ל- main.obj.

5. לאחר שלב הקומפילציה וה- link, תקבלו קובץ exe בשם main.exe .

תוכנית זו מקבלת פרמטר , והוא התוכנית בשפת miny , עבורה אנו בונים קומפיילר. לפיכך כדי להריץ את ה- main יש לכתוב :

c:\ > main name.txt (או כל שם אחר)

פלט תוכנית ה- main בשלב זה הוא הודעה בלבד והיא , האם בדיקת התחביר של תוכנית ה- miny הצליחה או לא. בהצלחה, תודפס הודעה מתאימה.

בכישלון, תודפס מספר השורה בתוכנית miny בה נמצאה השגיאה.

בנוסף תוכנית זו יוצרת את הקובץ tree.txt כפי שהוסבר לעיל , עם מבנה העץ המודפס.

במצב של כישלון יש לאתר את השגיאה, לתקן ולהריץ שוב.

המלצה ! – לפני בדיקת הקוד שלכם, מומלץ תמיד להריץ תוכנית זו על תוכנית הקלט שלכם כדי לוודא שהתחביר שלה נכון, ובמידה ובדיקה זו הצליחה, רק אז להריץ את התוכנית שלכם שלמעשה , תיצור p-code עבור תוכנית ה – miny .

6. כעת עליכם להתחיל סוף סוף לכתוב קוד בעצמכם.

- עליכם לכתוב תוכנית, שמממשת את הפונקציות Code,CodeR,CodeL וכו'. וכן להפעיל אותם על קלט מתאים.

פונקציות אלו מתרגמות תוכנית משפת miny לשפת p-code, כאשר עבורם התוכנית כולה (הקלט) יושבת בעץ (זה לא עץ בינרי).

- סריקת התוכנית היא למעשה סריקת העץ.
- כאמור, תרגום התוכנית לעץ, כבר נעשית עבורכם ע"י התוכנית שיצר ה- YACC (miny_tab.c) . לפיכך, כדי להשתמש בעץ כל שעליכם לעשות הוא להפעיל את הפונקציה המתאימה שתבנה את העץ , ולקבל מצביע לשורש העץ.

דוגמא להפעלת פונקציה זו וקבלת מצביע לשורש העץ, תוכלו למצוא בתוכנית main.c לפעול באופן דומה.

- מצביע לשורש העץ הנו מטיפוס NODE (צומת בעץ).
- מבנה צומת בעץ מוגדר בקובץ typedef.h ,וכן ניתן להכיר את השדות הרלוונטים ומספר הבנים בכל type בעזרת מבנה העץ המודפס בקובץ tree.txt.
- בתרגיל זה עליכם לממש פקודות השמה בלבד הכוללות גם פניות <u>למערכים סטטיים</u> (לא דינאמיים). וביטויים עם אופרטורים (+, - ,* , &,| וכדומה). תוכניות ה - miny שתריצו יכילו מבנה של תוכנית, כך שהפקודות בה הם פקודות השמה בלבד. כלומר כל תוכנית תצטרך להיות במבנה הבא :

PROGRAM name

(*אין צורך להגדיר משתנים בדידים, רק מערכים בכדי לדעת את הגודל שלהם *)

VAR a1: array[$l_1..u_1$, $l_2..u_2$, .. , $l_n..u_n$] ;

{

פקודות השמה

}

- עליכם לכתוב תוכנית ראשית ב – C שתקבל כפרמטר את תוכנית ה- miny לקימפול, ותשלב את כל האלמנטים הדרושים על מנת לתרגם את תוכנית ה- miny ל- p-code.

מבנה התוכנית שתכתבו :

- בניית עץ מתוכנית הקלט (בעזרת הפונקציה המתאימה)
- קבלת מצביע לשורש העץ.
- פתיחת קובץ output עבור ה- p-code.
- קריאה לפונקציה Code עם המצביע לשורש, לבניית ה-pcode

- סגירת קובץ ה- output.
- הדפסת קובץ ה- output (p-code).

תוכנית זו תהיה main ויש צורך להדר אותה עם ה-objects של כל התוכניות שקימפלתם כבר , מלבד ה – main.c כמובן שכן גם היא תוכנית main .

לסיום, הרצה של התוכנית שלכם עם הפרמטר – תוכנית ה- miny , אמורה להוציא כפלט קובץ p-code + הדפסת קובץ זה (קוד ה- p-code) למסך.

הנחיות :

עליכם לתרגם תכניות המכילות את ה- statements הבאים:

משפטי השמה הכוללים את האופרטורים הבאים:

NOT,|, &, >=, >,/= , ==,<= , <, % , /, *, -, + , FALSE ,TRUE

(גם מינוס אונרי)

```
#include <stdio.h>
#include <fcntl.h>
#include "typedef.h"
#include <malloc.h>
#include <string.h>
#include <stdlib.h>
#include "miny_tab.h"
#define max_add 31


void code(NODE r);
void codeL(NODE r);
void codeR(NODE r);
void initsimbol_t();
int getFromsimbol_t(char * str);
void insertIntosimbol_t(char *str);

typedef struct
      {
         char *valName;
         int address;
      } simbol_t;

//first adress start from 5
int adress = 5;

simbol_t my_simbol_t[max_add];

//  tree of tokens
FILE  *treefile;
//  Pmachine output
FILE *p_Output;

// a pointer to the program file
extern FILE *yyin;

// build the tree and make syntax checking
extern int yyparse (void);



NODE getTree(char *progFile)
```

```
{
      char *file;

      file = (char*) malloc (strlen(progFile)+1);
      strcpy(file,progFile);

      // yyin is an external variable that been used in yyparse as
pointer to the pearl
      // source file.
      if ((yyin = fopen(progFile,"r")) == NULL)
      {

         fprintf(stderr,"Error: file %s does not exist \n",file);

       exit(-1);
    }

      // yyparse is yacc (pearl_tab.c) function that parse over the
pearl program,
      // check syntax and build the program tree.
      yyparse();

      // root was initialized in yyparse while it was building the
tree.
      // root is the pointer of the returning tree.
      if (root==NULL)
        exit(-1);

      return(root);
}



int main()
{

 NODE theProgram;
 char buff[20];          //buff use for print the file at the end

 initsimbol_t();
 theProgram = getTree("ourTest.txt");
 treefile=fopen("tree.txt","w");
 print_tree(root,0);
 printf("\n");
 fclose(treefile);

 // Code P Machine
  p_Output=fopen("output.txt","w");
  code(root);
  fclose(p_Output);

  // open file output.txt and print to screen
  p_Output=fopen("output.txt","r");
  printf("\n\n\n\nprint the pMachine output file\n");
  while(fgets(buff, 5, p_Output)!=NULL)
      printf("%s", buff);


return (0);
}
```

```
////////////////////////////////////////////////////////////////////
////////////////////
////////////////////////////////////////////////////////////////////
////////////////////
void initsimbol_t()//this func init the temp simbol_t
{
      int i;
      for(i=0; i<max_add; i++)
        my_simbol_t[i].valName = NULL;
}
int getFromsimbol_t(char * str)// this func help get the add of str
{

      int i,j;
      j = 0;
      for(i=0;i<strlen(str);i++)
      {
            j = ((int)str[i])+j;
      }
      j = j % max_add;
            while(1)
            {
                  if(strcmp(my_simbol_t[j].valName,str) == 0)
                  {
                        return (my_simbol_t[j].address);
                  }
                  else
                        j=(j + 1)%max_add;
            }
}

void insertIntosimbol_t(char *str)// this fun search place to insert
in the symbol_t
{
      int i;
    int j;
      j = 0;
      for(i=0;i<strlen(str);i++)
      {
            j = j+((int)str[i]);
      }
      j=j%max_add;
      while(1)
      {
            if(my_simbol_t[j].valName == NULL)
            {
                  my_simbol_t[j].valName =
(char*)malloc(sizeof(char)*(strlen(str)+1));

                  strcpy(my_simbol_t[j].valName,str);
                  my_simbol_t[j].address = adress;

                  adress++;
                  break;
            }
            else
            {
                  j++;
                  j=j%max_add;
            }
```

```
	}
}

void code(NODE r)// this func take a node and do  the code_l and
code_r
{

	if (r!= NULL)
	{
		if(r->op == ASSIGN)
		{
			insertIntosimbol_t(r->s1->name);
			fprintf(p_Output,"Ldc a %d\n",getFromsimbol_t(r-
>s1->name));
			codeR(r->s2);	// right son
			fprintf(p_Output,"sto \n");
		}
		else code(r->s1);
	}
}

void codeL(NODE r)//find code_L
{
    insertIntosimbol_t(r->name);
	fprintf(p_Output,"Ldc a %d\n",getFromsimbol_t(r->name));
}

void codeR(NODE r)//find code_R
{
	if (r != NULL)
	{
		switch (r->op)
		{
			case ADD:		codeR(r->s1);
								codeR(r->s2);

	fprintf(p_Output,"Add\n");
								break;

			case DIV:		codeR(r->s1);
								codeR(r->s2);

	fprintf(p_Output,"Div\n");
								break;

			case AND:		codeR(r->s1);
								codeR(r->s2);

	fprintf(p_Output,"And\n");
								break;
			case IDE:

								insertIntosimbol_t(r-
>name);
								fprintf(p_Output,"Ldc
a %d\n",getFromsimbol_t(r->name));
								fprintf(p_Output,"Ind
i\n");

								break;
```

```
                case INTCONST:          fprintf(p_Output,"Ldc i %d
\n",r->num_val);
                                                break;

                case MUL:               codeR(r->s1);
                                                codeR(r->s2);

        fprintf(p_Output,"Mul\n");
                                                break;

                case OR:                codeR(r->s1);
                                                codeR(r->s2);

        fprintf(p_Output,"Or\n");
                                                break;

                case NOT:               codeR(r->s1);


        fprintf(p_Output,"Not\n");
                                                break;

                case EQU:           codeR(r->s1);
                                                codeR(r->s2);

        fprintf(p_Output,"Equ\n");
                                                break;

                case GRE:               codeR(r->s1);
                                                codeR(r->s2);

        fprintf(p_Output,"Gre\n");
                                                break;

                case GEQ:               codeR(r->s1);
                                                codeR(r->s2);

        fprintf(p_Output,"Geq\n");
                                                break;

                case LES:               codeR(r->s1);
                                                codeR(r->s2);

        fprintf(p_Output,"Les\n");
                                                break;

                case LEQ:               codeR(r->s1);


        fprintf(p_Output,"Leq\n");
                                                break;


                case MIN:               if(r->children==1)
                                                {
                                                        codeR(r->s1);

        fprintf(p_Output,"Neg\n");
                                                }
                                                else
                                                        {
```

```
                                                codeR(r->s1);
                                                codeR(r->s2);
        fprintf(p_Output,"Sub\n");
                                                }
                                                break;
            }
        }
}
```

שאלה 7:

1. נתון המבנה הבא שנקרא multicond. הפועל באופן הבא:

multicond (E1; E2; E3; .. ; En) { S1; S2; S3; .. ; Sn}

המבנה עובד באופן הבא:

בפעם הראשונה מחושב הערך של E1 . אם הוא True , אזי מבצעים את S1 , אחרת מסיימים את המבנה.

בפעם השניה מחושבים הערכים של E1 ו – E2 . רק אם שניהם True , אזי מבצעים את S2 , אחרת מסיימים את המבנה.

בפעם השלישית מחושבים הערכים של E1 ,E2 ו – E3. רק אם שלושתם True , אזי מבצעים את S3 , אחרת מסיימים את המבנה.

וכו..

בפעם האחרונה מחושבים הערכים של E1 ,E2 עד En. רק אם כולם True , אזי מבצעים את Sn , ואז מסתיים כל המבנה.

א. יש להסביר אילו שינויים אם בכלל נדרשים בכל אחד משלבי הקומפיילר :

מנתח לקסיקלי

_____

מנתח סינטקטי

_____ _____

מנתח סמנטי

_____

א. רשמו את הדקדוק של המבנה הנ"ל.

ב. יש לתת הסבר כללי על השינויים בקוד עבור המבנה לעיל .

ג. רשמו את הקוד עבור ה- pmachine באופן יעיל .

code (Multicond (E1; E2; E3; .. ; En) { S1; S2; S3; .. ; Sn}) ρ =

2. יש להוסיף למנתחים שקיבלתם את מבני ה-multicond וה- for ולערוך את כל השינויים הנדרשים.
3. יש לממש ב MINY את הפקודות הבאות:
   א. סדרת פקודות.
   ב. פקודות הסתעפות: IF-THEN-ELSE.

לולאות WHILE, FOR ו –MULTICOND .

<u>שאלה 8:</u>

רוצים להוסיף את הפקודה הבאה לשפת ה- miny.

double-while (B_1) (B_2) { S_1 } { S_2 }

כאשר E גוזר ביטוי בוליאני ו- S גוזר משפט או בלוק של משפטים.

משמעות הפקודה:

ביצוע S_1 כל עוד B_1 מתקיים וגם ביצוע S_2 כל עוד B_2 מתקיים. על פי התנאים הבאים:

4. כל עוד גם B_1 וגם B_2 מתקיימים, יש לבצע את S_1 ואחריו את S_2.
5. אם B_1 מפסיק להתקיים (ו-B_2 ממשיך להתקיים) יש לבצע את S_2 כל עוד B_2 מתקיים.
6. כנ"ל ההפך; כלומר, אם B_2 מפסיק להתקיים (ו-B_1 ממשיך להתקיים) יש לבצע את S_1 כל עוד B_1 מתקיים.

לאחר ש – B_1 הפסיק להתקיים, אין צורך לבדוק אותו שוב (כנ"ל לגבי B_2).

דוגמא:

```
x=1;
double-while ( x mod 5 <> 0)  // 5 – x לא מתחלק ב
             ( x mod 7 <> 0)  // 7 – x לא מתחלק ב
{        print("* %d", x);  x = x + 1;  }
{        print("# %d", x);  x = x + 1;  }
```

תוצאת ההדפסה:

* 1 # 2 * 3 # 4 # 5 # 6

א. יש להגדיר באופן מדויק וקצר (<u>בשורה או שתיים</u> יותר לא יתקבל!) אילו שינויים אם בכלל נדרשים בכל אחד משלבי הקומפיילר:

מנתח לקסיקלי _____double-while הוא אסימון חדש_____

_____

מנתח סינטקטי - (רק התוספת שנדרשת, יש לסמן אסימונים בקו תחתון)_____

__S → <u>double-while</u> <u>(</u> <u>_EXP_</u> <u>)</u> <u>(</u> <u>_EXP_</u> <u>)</u> <u>{</u> <u>S</u> <u>}</u> <u>{</u> <u>S</u> <u>}</u> _____

מנתח סמנטי __יש לבדוק עבור כל משתנה דקדוק את טיפוסו (EXP – בוליאני)_____

_____

(אם יש צורך במשתנים נוספים איך לקבוע אותם, ועל הרעיון על פיו אתם תצרו את הקוד)

ב. רשמו את הקוד עבור ה- pmachine :

code (double-while (B_1) (B_2) { S_1 } { S_2 }) ρ, nd =

```
L1: CodeR(B1) ρ, nd;
    fjp L3;
    CodeR(B2) ρ, nd;
    fjp L2;
    Code(S1) ρ, nd;
    Code(S2) ρ, nd;
    ujp L1;
L2: CodeR(B1) ρ, nd;
    fjp Lout;
    Code(S1) ρ, nd;
    ujp L2;
L3: CodeR(B2) ρ, nd;
    fjp Lout;
    Code(S2) ρ, nd;
    ujp L3;
Lout:
```

ג. הפעם עליכם לרשום <u>קוד יעיל</u>. ז"א יש לתרגם את B_1 ואת B_2 בקוד פעם אחת, וגם ה $Code(S_i)$ יתורגם פעם אחת בקוד.
מותר להגדיר משתני עזר (ולגשת אליהם מספר פעמים ככל שתרצו) שכתובתם תהיה בסוף השטח הסטאטי בפרוצדורה (וזה אומר שאם יש פקודה של double-while יש לזכור להוסיף ל –ssp עוד את סכום גדלי משתני העזר)

נגדיר שני משתנים בוליאניים במחסנית של הפרוצדורה בה מבוצעת הפקודה הזו – פנימיים של הקומפיילר _b1 ו- _b2

```
    CodeL(_b1); ldc b True; sto  b;    // Init with True
    CodeL(_b2); ldc b True; sto  b;

Lb1: CodeR(_b1) ρ, nd;
     fjp Lb2;
     CodeL( _b1) ρ, nd;       // Evaluate B1, store in _b1
     CodeR( B1) ρ, nd;
     sto b;

Lb2: CodeR(_b2) ρ, nd;
     fjp Lbb;
     CodeL( _b2) ρ, nd;       // Evaluate B2, store in _b2
     CodeR( B2) ρ, nd;
     sto b;

Lbb: CodeR(_b1) ρ, nd;
     CodeR(_b2) ρ, nd;
     or;
     fjp Lout;

Ls1: CodeR( _b1) ρ, nd;
     fjp Ls2;
     Code( S1) ρ, nd;

LS2: CodeR( _b2)  ρ, nd;
```

```
    fjp Lb1;
    Code( S2)  ρ, nd;

    ujp Lb1;

Lout:
```

שאלה 9:

רוצים להוסיף את הפקודה הבאה לשפת ה- miny.

$S \rightarrow$ do-max (L) S_1
$L \rightarrow COND$; L | COND
$COND \rightarrow BOOL$: EXP

משמעות הפקודה:

1. בתחילת הלולאה יש לעבור על התנאים שנגזרים מ L ע"פ הסדר שלהם בקוד. בכל תנאי COND צריך לבדוק את התנאי הבוליאני BOOL, ואם הוא מתקיים לחשב את EXP. בכל מקרה יש לעבור לתנאי הבא.
2. בסוף יש לבצע את S_1 מספר פעמים כערך הביטוי EXP המקסימלי שחושב בשלב הקודם ולצאת מהלולאה.
3. אם בשלב 1 אף ביטוי לא מתקיים, יש לצאת מהלולאה בלי לבצע את S_1.
4. אם בשלב 1 ערך הביטוי המקסימלי הוא קטן שווה ל-0, יש לצאת מהלולאה בלי לבצע את S_1.

דוגמא:

```
int a = 2, b = 1;
do-max (    a > b    :  a - 1)
            a > b    :  a
            a <= b   :  b + 2)
        print("do-max body once");
```

תוצאת ההדפסה:

```
do-max body once
do-max body once
```

ה. יש להגדיר באופן מדויק וקצר (בשורה או שתיים יותר לא יתקבל!) אילו שינויים אם בכלל נדרשים בכל אחד משלבי הקומפיילר:

מנתח לקסיקלי _____do-max הוא אסימון חדש, וגם : הם חדשים____

מנתח סינטקטי - (רק התוספת שנדרשת, יש לסמן אסימונים בקו תחתון) את כל הדקדוק שלמעלה

מנתח סמנטי __יש לבדוק שBOOL הוא בוליאני, ו- EXP הוא ביטוי מטיפוס מספר שלם.

(אם יש צורך במשתנים נוספים איך לקבוע אותם, ועל הרעיון על פיו אתם תצרו את הקוד)

ב. רשמו את הקוד עבור ה- pmachine :

הערות:
ניתן להניח שאין כבר שגיאות בקוד
שימוש ביותר משני משתני קומפיילר או שיערוך חוזר של אחד הביטויים יגררו ניקוד חלקי בלבד.

נגדיר שני משתנים בוליאניים במחסנית של הפרוצדורה בה מבוצעת הפקודה הזו – פנימיים של הקומפיילר _max ו-_v .

code (do-max (L) S_1) ρ, nd =

```
        CodeL(_max) ρ, nd; ldc i 0; sto i;
L1:     if(L → COND ; L) {        // אם עדיין יש עוד ביטויים נבדוק ונחזור לבא
                CodeR(BOOL) ρ, nd;    // נבדוק את הביטוי הבוליאני ואם הוא אמת נשערך את הערך
                fjp L1;
                CodeR(EXP) ρ, nd;
                dpl i;
                sro ρ(_v);                    // אם הערך גדול מאפס יש מה לבדוק
                ldc i 0;
                grt i;
                fjp L1;
                ldo ρ(_v);                    // אם הערך גדול מהמקסימום יש להציב אותו במקסימום
                ldo ρ(_max);
                grt i;
                fjp L1;
                ldo ρ(_v);
                sro ρ(_max);
                ujp L1;
                }
        if(L → COND) {        // אם אין יותר ביטויים נבדוק ונצא
                CodeR(BOOL) ρ, nd;
                fjp L2;
                CodeR(EXP) ρ, nd;
                dpl i;
                sro ρ(_v);
                ldc i 0;
                grt i;
                fjp L2;
                ldo ρ(_v);
                ldo ρ(_max);
                grt i;
                fjp L1;
                ldo ρ(_v);
                sro ρ(_max);
                }
L2:     ldo ρ(_max);     // יצירת קוד להרצת גוף הלולאה – מקס פעמים
        ldc i 0;
        grt i;
        fjp L3;
        code(S1) ρ, nd;
        ldo ρ(_max);
        ldc i 1;
        sub i;
        sro ρ(_max);
        ujp L2;
L3:
```

שאלה 10:

רוצים להוסיף לשפת miny את התכונה של נסיגה, דהיינו backtracking. כאשר הבטוי יהיה כדלקמן:

track(e_1, e_2, .. , e_n)
back

כאשר פקודת track תפורש באופן הלא פורמלי הבא. קודם ישוערך הבטוי e_1 , אם הבטוי e_1 יסתיים בהצלחה, אזי תוצאת הבטוי כולו תהיה התוצאה של שיערוך e_1. אם הבטוי e_1 נכשל, אזי משערכים את הבטוי הבא, e_2 . , אם הבטוי e_2 יסתיים בהצלחה, אזי תוצאת הבטוי כולו תהיה התוצאה של שיערוך e_2. אם הבטוי e_2 נכשל, אזי משערכים את הבטוי הבא, e_3 , וכה הלאה. התהליך ממשיך עד אשר הבטוי לא נכשל.

פקודת back גורמת לבטוי מסוג track להכשל. ז"א, כאשר back מבוצע, הוא גורם לשערוך ה e_i הבא אחריו (בתוך track כמובן).

בדוגמא הבאה, ישוערך הבטוי כולו ל- 3.

track({ 5; back}, back, 3, 7)

הסבר:

הבטוי הראשון משוערך ל – 5 ואז ה- back מכשיל אותו,
הבטוי השני מכיל רק back ולכן הוא נכשל,
הבטוי השלישי מכיל 3 ולכן כל הבטוי מחזיר 3,
הבטוי הרביעי לא ישוערך כיוון שהשלישי לא נכשל.

נתון הדקדוק הבא המכיל גם את הפקודה מסוג track:

E → if E then E else E fi
| while E do E od
| track(ELIST
| back
ELIST → E)
| E, ELIST

הסטודנטית ושתי אשר למדה קומפילציה טוענת כי הפקודה track איננה מוגדרת אם כל הבטויים אשר היא מכילה נכשלים. ושתי מציעה כי הפקודה track אסור שתכיל back כחלק מהבטוי האחרון, אלא אם כן back מוגדרת כבטוי המקונן בתוך הבטוי האחרון.

א. עזרו לושתי לשנות את הדקדוק כך שלא יכ ל פנייה ישירה ל – back כחלק מבטוי האחרון ב- track.

E → if E then E else E fi
| while E do E od
| track(ELIST
| back
ELIST → F)
| E, ELIST
F → if E then E else E fi
| while E do E od
| track(ELIST

ב. יש להסביר בשורה אחת בלבד אילו שינויים אם בכלל נדרשים בכל אחד משלבי הקומפיילר:

מנתח לקסיקלי ____שינוי של האסימונים – back ו - track____

מנתח סינטקטי _____הדקדוק של השינוי_____

מנתח סמנטי ___הבטוי של track צריך להיות מטיפוס מתאים לשאר_____

ג. יש לתת הסבר על השינויים בקוד Pmachine עבור התוספת לעיל.

```
code( track (ELSIT ) =  if (E) then code( E); not T; fjp L1; code( ELSIT)
                                else if (F) then code (F)
code ( E) = if ( if) then code (if)
            else if ( while) then code (while)
            else if ( track) then code (track)
            else if ( back) then code (back)
code( back) = false
code ( F) = if ( if) then code (if)
            else if ( while) then code (while)
            else if ( track) then code (track)
            ; L1:
```

ייצור קוד pmachine – פרוצדורות

שאלה 1:

```
program H;

   proc  A( proc F1; x:integer);

      proc  B ( proc F2);

         proc  C ( proc F3; z:integer );
          ssp __3___; sep ___8___;
          begin
             if (x+z > 0) then

(ssp = 2)        lda  2 7; ind i; lda  0 7; ind i; add i; ldc i 0; grt i; fjp l1;
                 begin
                 z := z - 1;

(ssp = 3)                lda  0 7; lda 0 7; ind i; ldc i 1 ; sub i; sto i;

                         F1( F3, z)

(ssp = 8)                mstf  2 6; lda a 0 5; movs 2; lda 7 0; ind i; smp 3; cupi 2 5;

                         end
          end;

      begin (*B*)
            x := x - 1;
            F2( C, x)
      end;

    begin (*A*)
         x := x - 1;
         B( F1)
   end;

begin (*H*)
   A( A , 4);
end.
```

א. יש לתרגם את הקוד של הפרוצדורה C כולל ה –ssp וה- sep.

יש להקפיד על ההוראות, אחרת התשובה לא תיבדק.

ב. יש להראות את מצב המחסנית עד לקריאה השניה ל - B.
אין צורך לפרט את המחסנית, אלא רק את ה – SL (ע"י חיצים מצד אחד), והמשתנים (x, z),
והפונקציות (F1, F2, F3) עליהן יש לרשום את ה SL של כל אחת לאורך העברת הפרמטרים).

C_2	Z = -3 F3 = C, SL(F3) = B_3	B_2
B_3	F2 = C, SL(F2) = B_2	A_3
A_3	X = ~~-1~~ ~~-2~~ -3 F1 = C, SL(F1) = B_2	H
C_1	Z = ~~0~~ -1 F3 = C, SL(F3) = B_2	B_1
B_2	F2 = C, SL(F2) = B_1	A_2
A_2	X=~~2~~ ~~1~~ 0 F1 = C, SL(F1) =B_1	H
B_1	F2 = A, SL(F2) = H	A_1
A_1	X = ~~4~~ ~~3~~ 2 F1 = A, SL(F1) = H	H
H		

ג. האם תכנית עוצרת? אם כן אחרי כמה קריאות?
נמקו – תשובה לא מנומקת לא תזכה בנקודות.

התכנית עוצרת כמו שמפורט במחסנית לעיל – אחרי הקריאה לC בפעם השניה.

שאלה 2:

```
program H;
   var a : array [-2..2] of integer;
       i :  integer;

   proc  A( z : array [0..i+1] of integer; var x : integer);
      var y : integer;

       proc  B ( proc T);
       begin
           T( i);
      end;

      func  C ( var y : integer ): integer;
           proc  D ( var z : integer );
           begin
                x:= x – 1;
                 C( z );
            end;

      begin (*C*)
            if ( y >= 3) then  D( y )
           else if (y>=2) then B( D)
      end;

    begin (*A*)
         i  := C(x );
   end;

begin (*H*)
   i := -2;
  while( i <= 2)
  begin
        a[i] := i;
        i := i + 1
  end;
  A( a , i);
end.
```

יש להראות את מצב המחסנית ברגע השיא של התכנית (כשהמחסנית בגובה המקסימלי שלה עבור התכנית הזו).
בתוך המחסנית יש לציין את כל הערכים (מלבד כתובת החזרה בקוד- סמנו אותם ב- RA).

אין לציין חיצים אלא מספרים ממש.
יש להתחיל במחסנית משמאל ואח"כ בימנית ,הכתובות מצוינות ליד!
ליד משתנים, מתארי מערכים (מערכים), SL, DL, EP- צריך לציין את שמותיהם בעמודה השמאלית - כדי שניתן יהיה לעקוד אחריהם.
אם ישנם ערכים שהשתנו תוך כדי ההרצה, יש למתוח על הערך הקודם קו, ולציין לידו את הערך החדש.

DL	16	36
SL	16	35
C		34
LV(i)	15	33
	2	32
	1	31
	0	30
	-1	29
	-2	28
y		27
x	15	26
a.d	4	25
a.d	0	24
a.d	0	23
a.d	5	22
a.d	28	21
	RA	20
EP	27	19
DL	0	18
SL	0	17
A		16
i	~~-2~~ ~~3~~ 1	15
A	2	14
A	1	13
A	0	12
A	-1	11
A	-2	10
a.d.	2	9
a.d.	-2	8
a.d.	-2	7
a.d	5	6
a.d	12 (10 - - 2)	5
	RA	4
EP	Null	3
DL	Null	2
SL	Null	1
H		0

		72
		71
		70
		69
y	15	68
	RA	67
EP	68	66
DL	57	65
SL	16	64
C		63
z	15	62
	RA	61
EP	64	60
DL	52	59
SL	46	58
D	(Addr. In CODE)	57
	RA	56
EP	58	55
DL	46	54
SL	16	53
B		52
y	15	51
	RA	50
EP	51	49
DL	40	48
SL	16	47
C		46
Z	15	45
	RA	44
EP	46	43
DL	34	42
SL	34	41
D		40
y	15	39
	RA	38
EP	38	37

שאלה 3:

לתרגם לpmachine

```
proc A( );
 var x : integer;
l0: ssp _6_ ; sep __6____; ujp l1;
                    proc B ( x : integer);
                   l2 : ssp __6__ ; sep ____3____; ujp l3;
                    begin
                    l3:
                     x := x + 1;
                       end;


                    proc C ( var y: integer );
                   l4: ssp ._6__. ; sep __7_; ujp l5;
                              proc D (proc G);
                                 l6: ssp ._7____. ; sep _6__; ujp l7;
                                        proc E ( var z : integer);
                                     l8: ssp __6__.. ; sep _7__; ujp l9;
                                        begin
                                            l9:x := 10;
                                         _lda a 4-1 5;_ldc i 10; sto i;
                                        G ( z + 2 );
                                 mstf 4-3 5; lda a 4-4 5; ind i ; ind i; ldc i 2; add i; smp 1; cupi 4-3 5;
                                         end;
                                 begin (*D*)
                               l7:
                                 E( y );
                                 _mst 3-3; lda a 3-2 5; ind i; cup 1 l8;
                                 end;
                    begin (*C*)
                   l5:
                     y := 6;
                     _lda a 0 5; ind i;_ldc i 6;_sto i;__
                     D(B);
                    _mst 0; ldc p l2; lda a 2-1 0; cup 2 l6;
                     end;
begin (*A*)
l1:
C( x );
_mst 0;_lda 0 5; cup 1 l4;_
end;
```

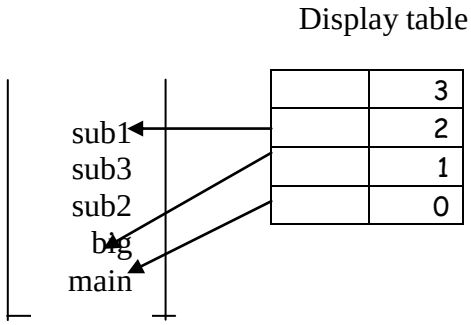
עשינו פעמיים ind i כי בפעם הראשונה זה כתובת ובשנייה ערך.

שאלה 4:

אלטרנטיבה אחרת לגישה לשרשרת הקריאות לפרוצדורות (incarnaton path) הינה לשמור את ה – static links במערך מחוץ למחסנית, שהכניסות אליו הן על פי רמת הקינון.
דוגמא (רמות קינון נתונות ע"פ האינדנטציה):

```
prog main
   proc big
       proc sub1
       proc sub2
   proc sub3
```

מחסנית זמן הריצה תהיה :

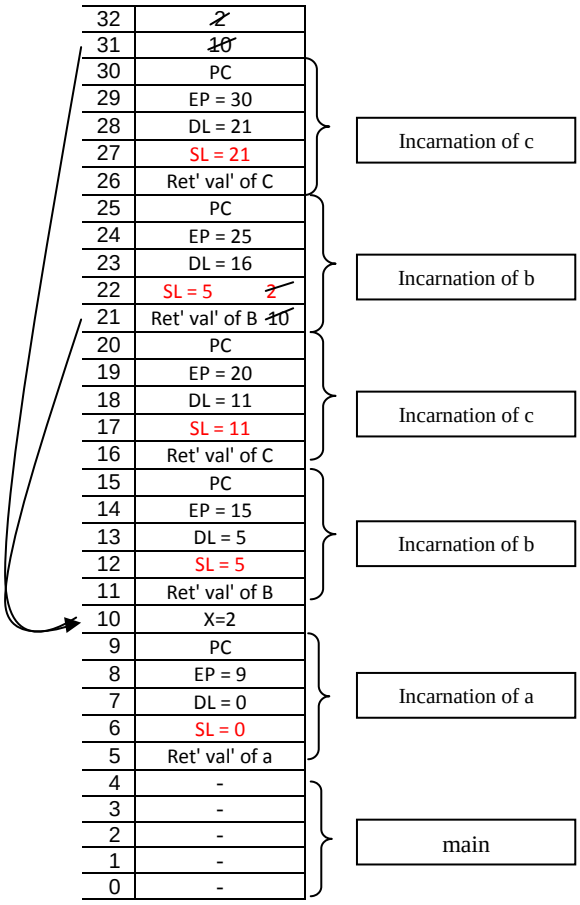


בטבלה עבור הפרוצדורה שעכשיו רצה – ישנם כניסות עבור כל רמה של קינון (nesting depth) ומצביעים עבור ה –incarnations של הפרוצדורות המתאימות (הקרובות ביותר) על פי רמת הקינון.
כך שאם מחפשים משתנה ברמת קינון 0 – נכנסים לטבלה ברמה 0 וכו'.

נתונה התכנית הבאה:

```
prog main;

   proc x;
   var a : integer;

      proc y;
         proc z;
         begin
            a := a + 1;
            y;
         end;
      begin  // y
         z;
      end;

  begin //x
     y;
     a = a * 2;
  end;

begin  // main
   a := 5;
   x;
end.
```

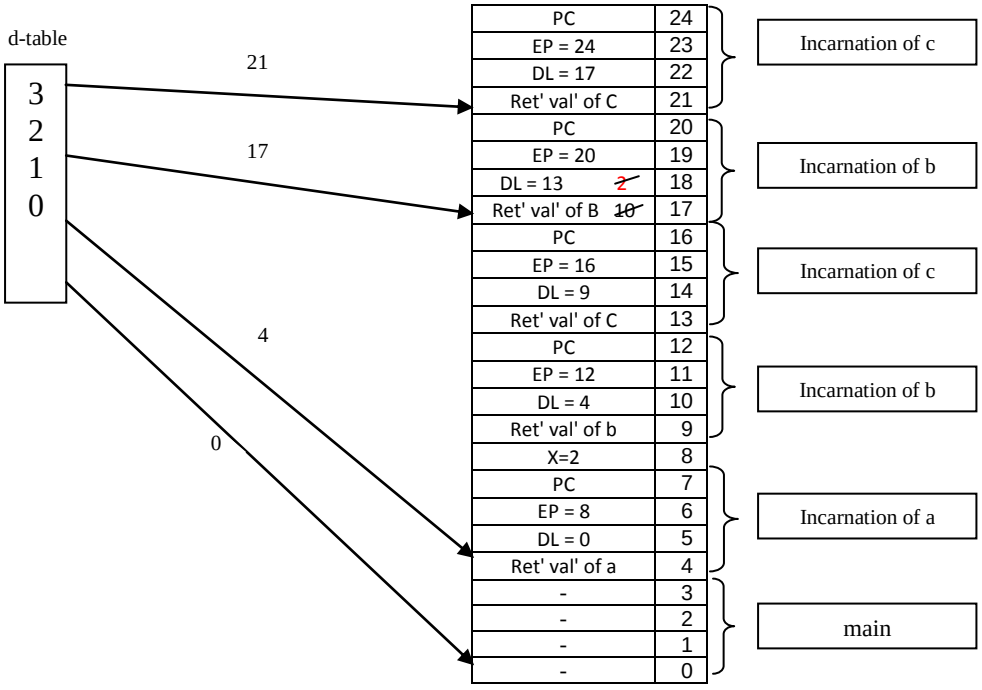
א. יש לתאר עבור התכנית מחסנית זמן ריצה עד הקריאה השלישית ל - z , לציין את ה – static links עבור כל incarnation ולציין את הכתובת המלאה של a (ρ(a עבור הקריאה השלישית של z.



P(x) = (2,5)

ב. יש לתאר עבור התכנית מחסנית זמן ריצה עד הקריאה השלישית ל - z , לציין את ה – display table עבור כל incarnation ולהסביר כיצד פונים ל - a עבור הקריאה השלישית של z.

מכיוון שנשתמש ב-display בשביל ה-static link נוריד את השורה של ה-static link מה-incarnation.
עבור הקריאה השנייה של x אנחנו נבצע את הדבר הבא: אנחנו נמצאים ברמת קינון 3 ורמת הקינון של x בשימוש היא 4 ובהגדרה היא 2.
לכן החישוב יתבצע כך: נבצע חיסור בין רמת הקינון הנוכחית לבין ההפר ש בין רמת הקינון של x בשימוש לבין ההגדרה.
נסמן: רמת קינון x בשימוש = nd_x.
רמת קינון x בהגדרה = nd'_x.
רמת קינון נוכחי = d.
לכן: ההפרש: $d-(nd_x-nd'_x)$ הוא בדיוק היכן שמוגדר x, מכיוון שכל incarnation עכשיו הוא 4 שורות בטבלה, נקפוץ 4 תאים ונגיע לכתובת של x.
לכן , p(x) = 8.



ג. תחת אילו תנאים כלליים שרשרת ה – static links למציאת משתנים לא מקומיים תיתן ביצועים חלשים יותר מאשר טכניקת ה – display?

יש להניח שלא מתבצעות שום אופטימיזציות על הקוד, וכמו כן אין העברת פרוצדורות כפרמטרים.

בהנחה שלא מתבצעות שום אופטימיזציות על הקוד ואין העברת פרוצדורות כפרמטרים אזי נובע שתמיד שרשרת ה- static link למציאת משתנים לא מקומיים תניב ביצועים חלשים יותר מטכניקת ה- display וזאת מכיוון ששרשרת ה-static link יותר איטית, מכילה חישובים, מה שלא קיים בשיטת ה-display.
כמו כן הדבר מתבטא גם בסיבוכיות מציאת הכתובת, למשל, במצב שאנחנו נמצאים ברמת קינון n ומשתמשים במשתנה לא מקומי המוגדר ברמת קינון 0, שרשרת ה-static link תמצא את כתובתו של המשתנה בסיבוכיות זמן ריצה של $O(n)$ כי היא תצטרך לבצע n קפיצות אחורה, בניגוד ל- display שתמצא את כתובתו של המשתנה ב-$O(1)$.

שאלות נוספות + אופטימיזציות

א. הטווח של פונקציות ב- c הינו מרגע הגדרת הפונקציה ועד סוף הקובץ.
כיצד עובדה זו מקלה על הקומפיילר?

כאשר הקומפיילר בודק את תאימות הקריאה להגדרה – הנתונים כבר נמצאים בטבלת הסמלים – אין צורך ב- backtracking__

ב. "קליק" הינו גרף שבו ישנה קשת בין כל שני צמתים. נתון interference graph שהוא "קליק" עם k צמתים. האם ניתן לצבוע אותו ב- k צבעים ? אם לא נמק, אם כן, כמה אפשרויות צביעה ישנן?

____כן, כי לכל צומת יש k-1 שכנים, ואז ניתן להסיר ולצבוע.___

____מספר האפשרויות הינו k!_____

ג. מהו דקדוק דו-משמעי?

____דקדוק שבו לכלל גזירה יש יותר מבחירה אחת – שני עצי גזירה שונים_____

ד. נתונה התכנית הבאה:

```
prog  main;
var y;

        proc p1;
        var y;
       {  y := 1;  p2 }

        Proc p2;
        { print y }

{ y:=2;  p1; }
```

A. מה תהיה תוצאת ההדפסה עבור static binding :

2

B. מה תהיה תוצאת ההדפסה עבור dynamic binding :

1

שאלה 2:

נתונה הפונקציה הבאה ב- C :

```
void f( int arrA[], int arrB[],  int nA, int nB, int i)
{         arrA[2 * i] = (nA + 1) * nB;
          nB = 5 * i;
          arrB[nA + 1]  = arrB[2 * i] + nA * nB;
}
```

ונתון תרגום לקוד ביניים:

```
Ro = nA + 1
R1 = R0 + nB
R2 = 2 * i
Store arrA[R2] = R1
nB = 5 * i
R3 = 2 * i
R4 = Load arrB[R3]
R5 = nA * nB
R6 = R4 + R5
R7 = nA + 1
Store arrB[R7] = R6
```

בהינתן הקוד לעיל, יש ליצור את ה- interfernce graph עבור המשתנים והרגיסטרים הוירטואליים, למעט המערכים. כיוון שהגרף גדול, יש לכתוב עבור כל משתנה או רגיסטר את שכניו בטבלה הבאה – יש לסמן '1' במקום בו ישנה קשת מהמשתנה הרשום בעמודה השמאלית, לבין כל אחד מהאחרים

Edge to->	nA	nB	i	R0	R1	R2	R3	R4	R5	R6	R7
nA		1	1	1	1	1	1	1	1	1	
nB	1		1	1			1	1			
i	1	1		1	1	1					
R0	1	1	1								
R1	1		1			1					
R2	1		1		1						
R3	1	1									
R4	1	1							1		
R5	1	1						1			
R6	1										1
R7										1	

ב. האם ניתן לצבוע את הגרף שנוצר ב- 4 צבעים ? יש להסביר מדוע.
אין לצייר את הגרף ולהפעיל עליו את האלגוריתם . **יש להסביר** ע"פ התוצאות בטבלה מדוע התשובה היא כמו שבחרתם.
תשובה לא מנומקת לא תתקבל.

לכל הצמתים מלבד nA, nB, i יש פחות מארבעה שכנים, לכן אין בעייה לצבוע אותם, ואחרי שמסירים אותם מהגרף, ונשארים שלושה צמתים, לא נשארת בעייה.

ג. מפעילים אופטימיזציה על הקוד, שבה ביטויים שחושבו כבר לא יחושבו שנית, ומשתמשים בתוצאה של החישוב הקודם. כיצד יראה הקוד עכשיו?
יש למחוק פקודות מיותרות, ולהחליף שמות משתנים אם יש צורך. פקודה שנמחקת יש להעביר עליה קו, ולהעתיק לצד ימין רק פקודות שתכנן השתנה (**אין להעתיק את כל הפקודות שוב!!)**

Ro = nA + 1
R1 = R0 + nB
R2 = 2 * i
Store arrA[R2] = R1
nB = 5 * i
~~R3 = 2 * i~~
R4 = Load arrB[~~R3]~~ R4 = Load arrB[R2]
R5 = nA * nB
R6 = R4 + R5
~~R7 = nA + 1~~
Store arrB[~~R7] =~~ R6 Store arrB[R0] = R6

ד. תוך עיון בקוד החדש שנוצר, כיצד ישפיע על הגרף מהסעיף הראשון. האם עכשיו ניתן לצבוע את הגרף ב 4 צבעים.
יש לנמק את התשובה, תשובה לא מנונמקת לא תתקבל!

עכשיו הגדלנו את אורך החיים של R0, R2 וזה אומר שמספר הקשתות גדל – אפשר לראות שלשני המשתנים הללו יש 4 שכנים – לכן בלתי אפשר לצבוע ב- 4 צבעים

<u>שאלה 3:</u>
נתון קטע הקוד הבא:

```
b := x  * 0
a := a + 1
n := b
g := a + n
u := b + n
y := b * 2
g = b * b
e := y + u
s := 2 + 10
```

בנוסף, ידוע כי המשתנים החיים בסוף הקטע הנ"ל הם: **e, s**.

א. מלאו את הטבלה הבאה: (def, use וטווח חיים)

the code	**def**	**use**	**liveout**
b := x * 0	b	x	b
a := a + 1	b	a	b, a
n := b	n	b	b, a, n
g := a + n	g	a, n	b, n
u := b + n	u	b, n	u, b
y := b * 2	y	b	u, b, y
g = b * b	g	b	u, y
e := y + u	e	y, u	e
s := 2 + 10	s	-	e, s

א. מלאו את מטריצת הסמיכויות הבאה עבור אלגוריתם צביעת הגרף:

	b	x	a	n	g	u	y	e	s
b			1	1		1	1		
x									
a	1			1					
n	1		1						
g									
u	1						1		
y	1					1			
e									1
s								1	

ב. האם ניתן לצבוע את הגרף בשלושה צבעים? אם כן, הראו את הדרך ותנו את הקוד הסופי בשלושה רגיסטרים.
אם לא, הראו את הדרך ונמקו מדוע לא ניתן.

כן ניתן.
רואים שחוץ מ b לכולם יש פחות משלושה שכנים, לכן מתחילים להוריד מהשאר ואז ל-b יורדים קשתות ונשאר בשבילו רגיסטר.

```
R2 := R0  * 0
R1 := R1 + 1
R0 := R2
R1 := R1 + R0
R1 := R2 + R0
R0 := R2 * 2
R0 = R2 * R2
R1 := R0 + R1
R0 := 2 + 10
```