2א)

נראה עץ עבור הביטוי הראשון :

לא ניתן להראות עץ עבור הביטוי השני , שכן ע"פ כללי הדקדוק, סוגריים מסולסלים תמיד יכילו בתוכם ערכים שמופרדים ב ":" או סוגריים ריקים, ז.א לא ניתן להגיע לערך סופי ללא הפרדה של ":" .

כיוון שהגזירה של members->keyvalue->string:string/string:int/string:obj

כאשר הגזירה של obj היא ל members או לסוגריים ריקים ( ואז חוזר חלילה ..)

3)

נראה שתי עצי גזירה שונים עבור הביטוי :

בדומה למה שראינו בשיעור ניתן לבצע את הכלל

members->members,members בחלקים שונים בעץ הגזירה ( פעם בחלק שמאל של תת העץ ופעם בחלק ימין של תת העץ).

עץ גזירה ראשון :

עץ גזירה שני :

**Q4**

1. The SELECT sets intersect in these cases:

* *obj -> LB RB*
* *obj -> LB members RB*
* (Both of the sets have “LB”.)
* *members -> keyvalue*
* *members -> members COMMA members*
* (Both of sets have “string”.)

Because of these intersections in SELECT sets, this grammar is not LL(1) .

b+c. See in grammar\_output.txt, we created an LL(1) grammar for the language that answers both sections b and c.

7)

ניתן לתאר כול שפה רגולרית ע"י ביטוי רגולרי מהאותיות ∑ ופעולות: ∙,+,\* ( שרשור איחוד klenee star

לכן מספיק להוכיח עבור ביטוי רגולרי עם מבנה המכיל את הפעולות הנ"ל .

נוכיח באינדוקציה על מבנה הביטוי .

בסיס :

עבור אות בודדת a מ ∑ נבנה את הדקדוד .S->a

צעד: נניח נכונות עבור ביטוי רגולרי כולשהו A,B.

עבור : A+B מהנחת האינדוקציה קיים איזושהו דקדוק S1->A וכן קיים דקדוק S2->B ולכן הדקדוק

S->S1|S2 הינו דקדוק מתאים .

עבור: A∙B מהנחת האינדוקציה קיים איזושהו דקדוק S1->A וכן קיים דקדוק S2->B ולכן הדקדוק

S->S1S2 הינו דקדוק מתאים .

עבור A\* מהגדרה קיים 0<=Kכך ש \*לכן מהנחת האינדוקציה קיימים K דקדוקים שונים S1->X1,S2->X2…SK->XK לכן הדקדוק S->S1S2…SK הינו הדקדוק המבוקש.