בשביל קובץ part2 היה עלינו לייצר קובץ parser לפי הדקדוק שניתן לנו בתרגיל. את חוקי הגזירה בנינו לפי הדקדוק הנתון (הבן של צומת הוא הסימבול או הטרמינל השמאלי ביותר, וכל האחרים משורשרים ל"בן" על ידי פונקציית concate הנתונה), כאשר את סדר הופעת הטרמינלים היינו צריכים לסדר לפי התיעדופים הנדרשים ואת האסוציאטיביות לפי האסוציאטיביות הנדרשת משפת C. למשל העדיפות של AND גבוה יותר משל OR ולכן נמצא מתחתיו ברשימת הטרמינלים, והעדיפות של MULOP גבוה משל ADDOP ולכן גם הוא נמצא מתחתיו ברשימת הטרמינלים. דבר נוסף לדאוג לו הוא האסוציאטיביות. לטרמינלים כמו פעולות חשבון שיכולות להשתרשר בזו אחר זו ולסדר החישוב יש משמעות חייבת להיות אסוציאטיביות מוגדרת למשל 5+1-1: נרצה שהparser יחשב קודם כך (1-1)+5=5 ולא כך: 1-(1+5)=-5. לכל פעולה כזו נדרש אסוציאטיביות שמתבטא בקידומת %left שמגדיר אסוציאטיביות שמאלית, או %precedence כשאני רוצה שכלל א' יהיה יותר חזר מכלל ב'.

קובץ נוסף שהיה עלינו לשנות הוא קובץ lex. את קובץ הlex שינינו על ידי כך שכל "מציאה" של ביטוי רגולרי שיתאים לטרמינל ייצר node חדש עבור הטרמינל, שיעבור לparser (שמייצר צמתים לסימבולים) בתור הnode של הטרמינל המתאים בכלל הגזירה שעכשיו גוזרים. בנוסף lex יחזיר את האסימון שנמצא, למשל ID, WRITE, READ וכו'. גם פה, היה עלינו לסדר את הביטויים הרגולרים לפי תיעדופים.

התוספת האחרונה שביצענו הייתה הגדרת yylval מחדש על מנת שיוכל לקבל משתנה *PasrerNode ולכן הוספנו את השורה *#define YYSTYPE ParserNode בקובץ part2_helpers.h.