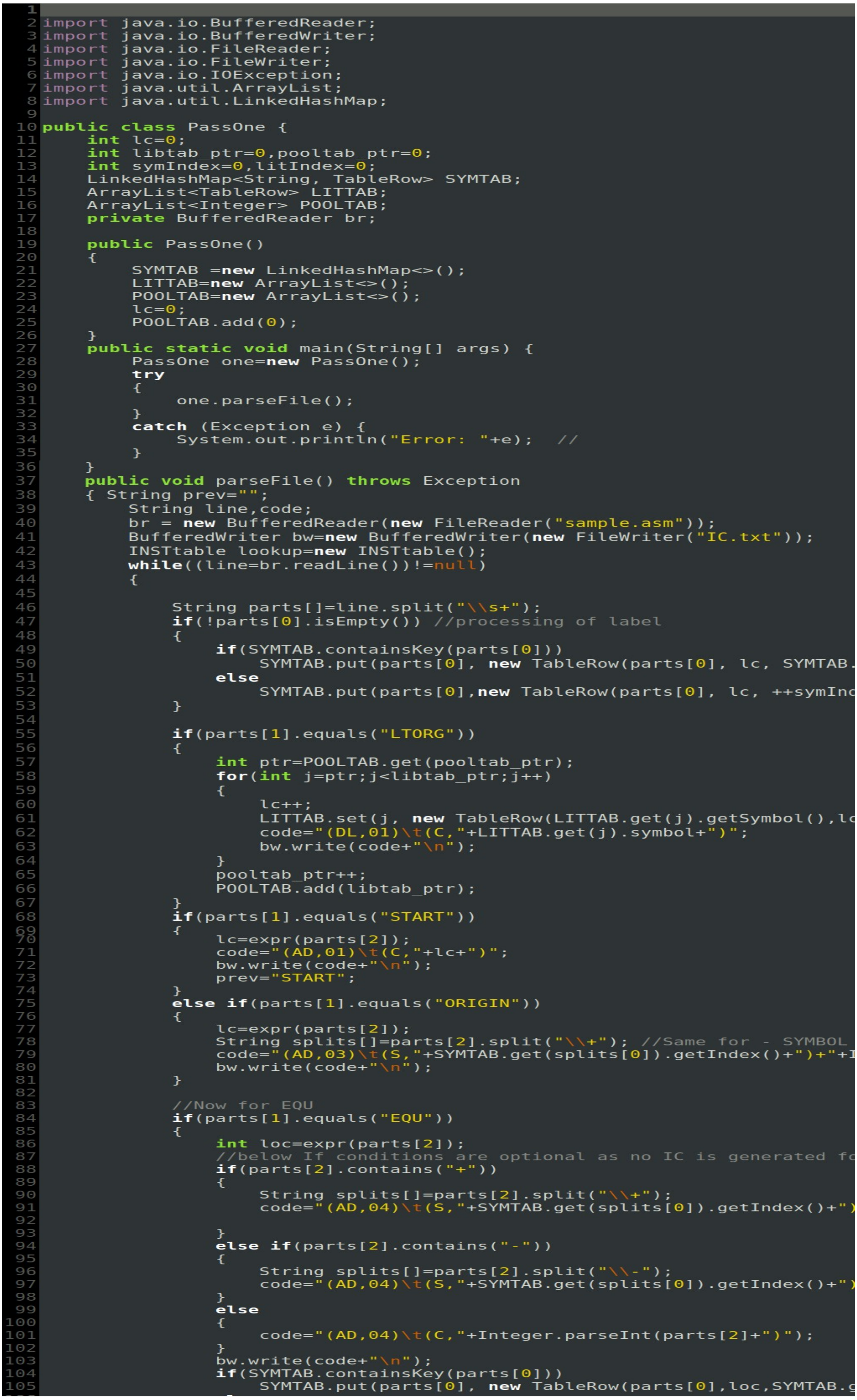
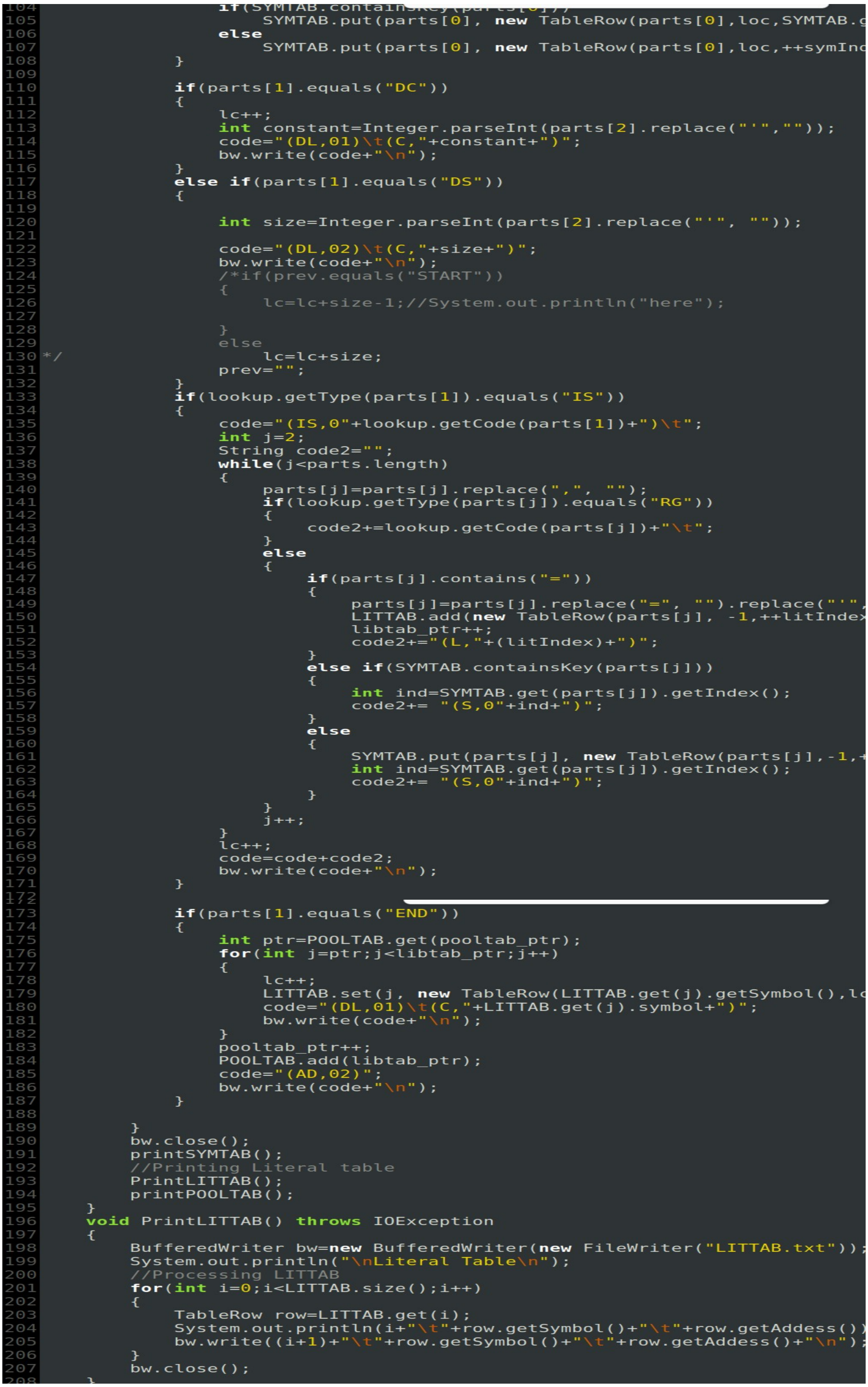
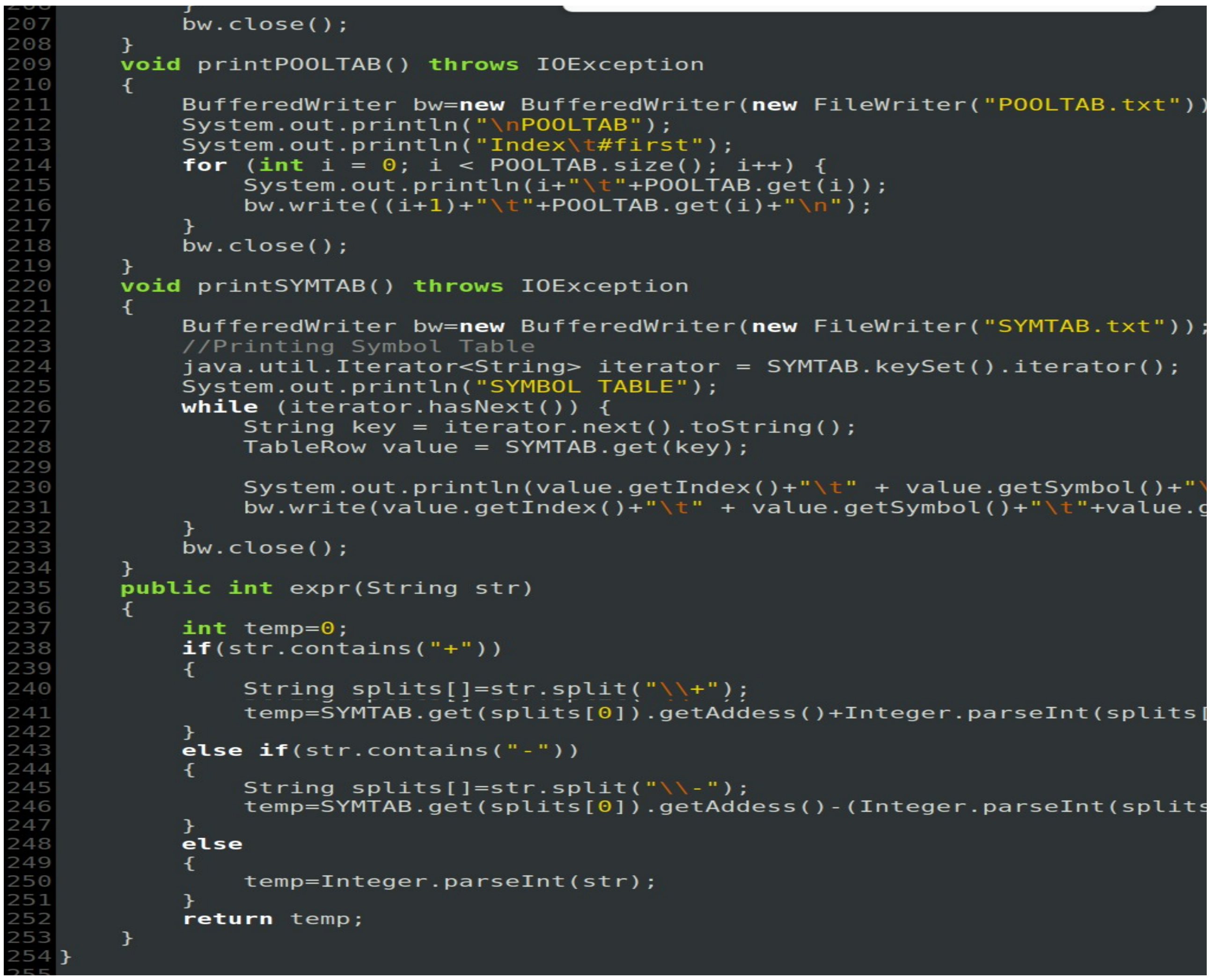


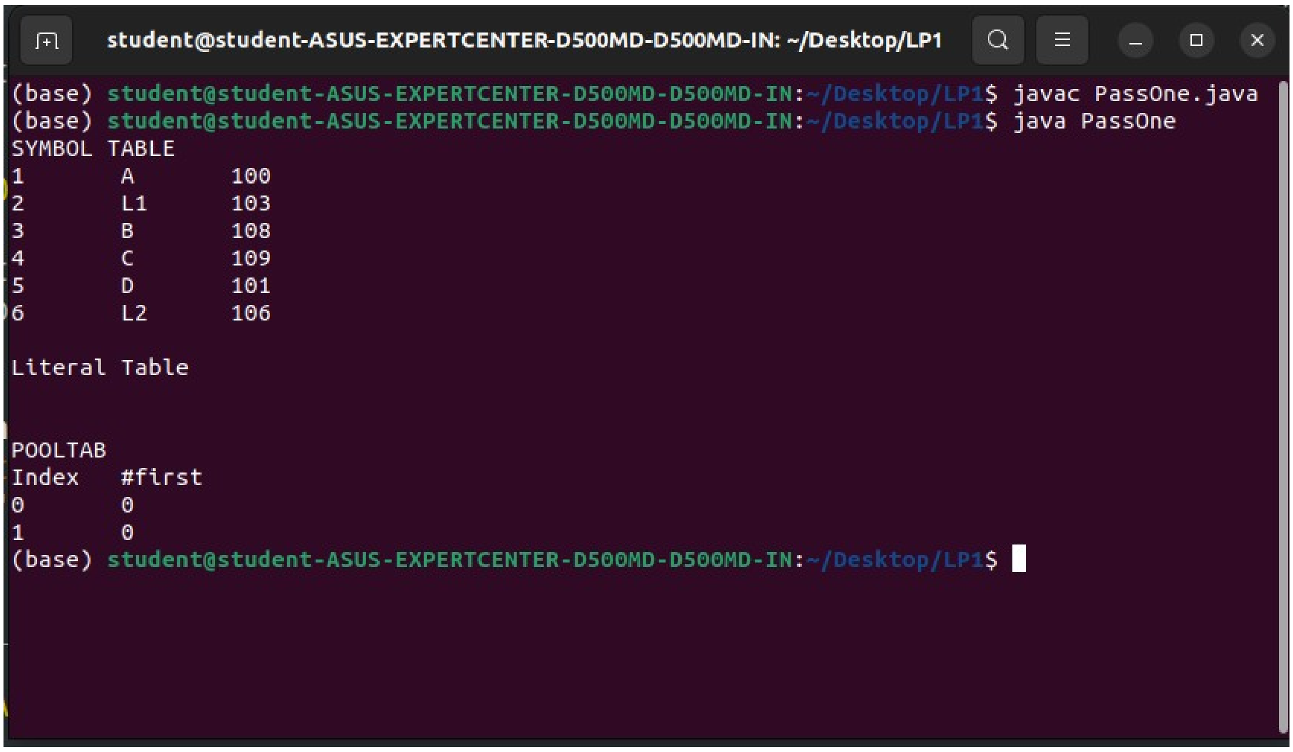
```java

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.LinkedHashMap;

public class PassOne {
    int lc=0;
    int libtab_ptr=0,pooltab_ptr=0;
    int symIndex=0,litIndex=0;
    LinkedHashMap<String, TableRow> SYMTAB;
    ArrayList<TableRow> LITTAB;
    ArrayList<Integer> POOLTAB;
    private BufferedReader br;

    public PassOne()
    {
        SYMTAB =new LinkedHashMap<>();
        LITTAB=new ArrayList<>();
        POOLTAB=new ArrayList<>();
        lc=0;
        POOLTAB.add(0);
    }
    public static void main(String[] args) {
        PassOne one=new PassOne();
        try
        {
            one.parseFile();
        }
        catch (Exception e) {
            System.out.println("Error: "+e);  //
        }
    }
    public void parseFile() throws Exception
    { String prev="";
        String line,code;
        br = new BufferedReader(new FileReader("sample.asm"));
        BufferedWriter bw=new BufferedWriter(new FileWriter("IC.txt"));
        INSTtable lookup=new INSTtable();
        while((line=br.readLine())!=null)
        {

            String parts[]=line.split("\\s+");
            if(!parts[0].isEmpty()) //processing of label
            {
                if(SYMTAB.containsKey(parts[0]))
                    SYMTAB.put(parts[0], new TableRow(parts[0], lc, SYMTAB.
                else
                    SYMTAB.put(parts[0],new TableRow(parts[0], lc, ++symInc
            }

            if(parts[1].equals("LTORG"))
            {
                int ptr=POOLTAB.get(pooltab_ptr);
                for(int j=ptr;j<libtab_ptr;j++)
                {
                    lc++;
                    LITTAB.set(j, new TableRow(LITTAB.get(j).getSymbol(),lc
                    code="(DL,01)\t(C,"+LITTAB.get(j).symbol+")";
                    bw.write(code+"\n");
                }
                pooltab_ptr++;
                POOLTAB.add(libtab_ptr);
            }
            if(parts[1].equals("START"))
            {
                lc=expr(parts[2]);
                code="(AD,01)\t(C,"+lc+")";
                bw.write(code+"\n");
                prev="START";
            }
            else if(parts[1].equals("ORIGIN"))
            {
                lc=expr(parts[2]);
                String splits[]=parts[2].split("\\+"); //Same for - SYMBOL
                code="(AD,03)\t(S,"+SYMTAB.get(splits[0]).getIndex()+")+"+I
                bw.write(code+"\n");
            }

            //Now for EQU
            if(parts[1].equals("EQU"))
            {
                int loc=expr(parts[2]);
                //below If conditions are optional as no IC is generated fo
                if(parts[2].contains("+"))
                {
                    String splits[]=parts[2].split("\\+");
                    code="(AD,04)\t(S,"+SYMTAB.get(splits[0]).getIndex()+")

                }
                else if(parts[2].contains("-"))
                {
                    String splits[]=parts[2].split("\\-");
                    code="(AD,04)\t(S,"+SYMTAB.get(splits[0]).getIndex()+")
                }
                else
                {
                    code="(AD,04)\t(C,"+Integer.parseInt(parts[2]+")");
                }
                bw.write(code+"\n");
                if(SYMTAB.containsKey(parts[0]))
                    SYMTAB.put(parts[0], new TableRow(parts[0],loc,SYMTAB.g
```



```
                if(SYMTAB.containsKey(parts[0]))
                    SYMTAB.put(parts[0], new TableRow(parts[0],loc,SYMTAB.g
                else
                    SYMTAB.put(parts[0], new TableRow(parts[0],loc,++symInc
            }

            if(parts[1].equals("DC"))
            {
                lc++;
                int constant=Integer.parseInt(parts[2].replace("'",""));
                code="(DL,01)\t(C,"+constant+")";
                bw.write(code+"\n");
            }
            else if(parts[1].equals("DS"))
            {

                int size=Integer.parseInt(parts[2].replace("'", ""));

                code="(DL,02)\t(C,"+size+")";
                bw.write(code+"\n");
                /*if(prev.equals("START"))
                {
                    lc=lc+size-1;//System.out.println("here");

                }
                else
*/                  lc=lc+size;
                prev="";
            }
            if(lookup.getType(parts[1]).equals("IS"))
            {
                code="(IS,0"+lookup.getCode(parts[1])+")\t";
                int j=2;
                String code2="";
                while(j<parts.length)
                {
                    parts[j]=parts[j].replace(",", "");
                    if(lookup.getType(parts[j]).equals("RG"))
                    {
                        code2+=lookup.getCode(parts[j])+"\t";
                    }
                    else
                    {
                        if(parts[j].contains("="))
                        {
                            parts[j]=parts[j].replace("=", "").replace("'",
                            LITTAB.add(new TableRow(parts[j], -1,++litIndex
                            libtab_ptr++;
                            code2+="(L,"+(litIndex)+")";
                        }
                        else if(SYMTAB.containsKey(parts[j]))
                        {
                            int ind=SYMTAB.get(parts[j]).getIndex();
                            code2+= "(S,0"+ind+")";
                        }
                        else
                        {
                            SYMTAB.put(parts[j], new TableRow(parts[j],-1,+
                            int ind=SYMTAB.get(parts[j]).getIndex();
                            code2+= "(S,0"+ind+")";
                        }
                    }
                    j++;
                }
                lc++;
                code=code+code2;
                bw.write(code+"\n");
            }

            if(parts[1].equals("END"))
            {
                int ptr=POOLTAB.get(pooltab_ptr);
                for(int j=ptr;j<libtab_ptr;j++)
                {
                    lc++;
                    LITTAB.set(j, new TableRow(LITTAB.get(j).getSymbol(),lc
                    code="(DL,01)\t(C,"+LITTAB.get(j).symbol+")";
                    bw.write(code+"\n");
                }
                pooltab_ptr++;
                POOLTAB.add(libtab_ptr);
                code="(AD,02)";
                bw.write(code+"\n");
            }

        }
        bw.close();
        printSYMTAB();
        //Printing Literal table
        PrintLITTAB();
        printPOOLTAB();
    }
    void PrintLITTAB() throws IOException
    {
        BufferedWriter bw=new BufferedWriter(new FileWriter("LITTAB.txt"));
        System.out.println("\nLiteral Table\n");
        //Processing LITTAB
        for(int i=0;i<LITTAB.size();i++)
        {
            TableRow row=LITTAB.get(i);
            System.out.println(i+"\t"+row.getSymbol()+"\t"+row.getAddess())
            bw.write((i+1)+"\t"+row.getSymbol()+"\t"+row.getAddess()+"\n");
        }
        bw.close();
    }
```



```java
        bw.close();
    }
    void printPOOLTAB() throws IOException
    {
        BufferedWriter bw=new BufferedWriter(new FileWriter("POOLTAB.txt"))
        System.out.println("\nPOOLTAB");
        System.out.println("Index\t#first");
        for (int i = 0; i < POOLTAB.size(); i++) {
            System.out.println(i+"\t"+POOLTAB.get(i));
            bw.write((i+1)+"\t"+POOLTAB.get(i)+"\n");
        }
        bw.close();
    }
    void printSYMTAB() throws IOException
    {
        BufferedWriter bw=new BufferedWriter(new FileWriter("SYMTAB.txt"));
        //Printing Symbol Table
        java.util.Iterator<String> iterator = SYMTAB.keySet().iterator();
        System.out.println("SYMBOL TABLE");
        while (iterator.hasNext()) {
            String key = iterator.next().toString();
            TableRow value = SYMTAB.get(key);

            System.out.println(value.getIndex()+"\t" + value.getSymbol()+"
            bw.write(value.getIndex()+"\t" + value.getSymbol()+"\t"+value.g
        }
        bw.close();
    }
    public int expr(String str)
    {
        int temp=0;
        if(str.contains("+"))
        {
            String splits[]=str.split("\\+");
            temp=SYMTAB.get(splits[0]).getAddess()+Integer.parseInt(splits[
        }
        else if(str.contains("-"))
        {
            String splits[]=str.split("\\-");
            temp=SYMTAB.get(splits[0]).getAddess()-(Integer.parseInt(splits
        }
        else
        {
            temp=Integer.parseInt(str);
        }
        return temp;
    }
}
```



student@student-ASUS-EXPERTCENTER-D500MD-D500MD-IN: ~/Desktop/LP1

```
(base) student@student-ASUS-EXPERTCENTER-D500MD-D500MD-IN:~/Desktop/LP1$ javac PassOne.java
(base) student@student-ASUS-EXPERTCENTER-D500MD-D500MD-IN:~/Desktop/LP1$ java PassOne
SYMBOL TABLE
1       A       100
2       L1      103
3       B       108
4       C       109
5       D       101
6       L2      106

Literal Table


POOLTAB
Index   #first
0       0
1       0
(base) student@student-ASUS-EXPERTCENTER-D500MD-D500MD-IN:~/Desktop/LP1$
```