Name: Sejal Girish Mahajan

Roll no. 05

**PASS 2**

```
import re
pattern = re.compile(r'\(\'([A-Z]{,2})\',\s(\d+)\)')

tab = "\t"

class pass2:
    def __init__(self):
        self.ICFile = open("intermediateCode.txt", mode='r')
        self.literalTableFile = open("literalTable.txt",mode='r')
        self.symboltableFile = open("symbolTable.txt",mode='r')
        self.outputFile = open("output.txt",mode='w')
        self.literalTable = {}
        self.symbolTable = {}

    def convertToString(self,string):
        string = str(string)
        if len(string) == 1:
            return "00" + string
        elif len(string) == 2:
            return "0" + string
        elif len(string) == 3:
            return string

    def readSymbolTable(self):
        print ("\nSymbol Table:")
        for line in self.symboltableFile.readlines():
            line = line.split("\t")
            index = int(line[0])
        location = int(line[2])
        self.symbolTable[index] = location
        print(index,location,sep = "\t")
    print("\n")

def readLiteralTable(self):
    print("\nLiteral Table:")
    for line in self.literalTableFile.readlines():
        line = line.split('\t')
        index = int(line[0])
        location = int (line[2])
        self.literalTable[index] = location
        print(index,location, sep = "\t")
    print("\n")

def parseFile(self):
    self.readLiteralTable()
    self.readSymbolTable()
    print("Machine Code:")
    print("LC\tOPCODE\tOP1\tOP2")
    for line in self.ICFile.readlines():
        line = line.strip("\n")
        line = line.split("\t")
        find = pattern.search(line[0])

        if find.group(1) == "IS" or find.group(1) == "DL":
            lineToParse = ""
            location = line[-2]
            lineToParse += location + tab

            if find.group(1) == "IS":
```

```
        lineToParse += self.convertToString(find.group(2)) +tab

        if find.group(2) == "10" or find.group(2) == "9":
            find = pattern.search(line[1])
            key = int(find.group(2))
            lineToParse += "000" + tab +self.convertToString(self.symbolTable[key]) + "\n"
        elif find.group(2) == "0":
            lineToParse += "000" + tab + "000" + "\n"

        else:
            find = pattern.search(line[1])
            lineToParse += self.convertToString(find.group(2)) + tab

            find = pattern.search(line[2])
            if find.group(1) == "S":
                key = int(find.group(2))
                lineToParse += self.convertToString(self.symbolTable[key]) + "\n"
            elif find.group(1) == "L":
                key = int(find.group(2))
                lineToParse += self.convertToString(self.literalTable[key]) + "\n"

    else:
        if find.group(2) == "1":
            lineToParse += "000" + tab + "000" + tab
            find = pattern.search(line[1])
            lineToParse += self.convertToString(find.group(2)) + "\n"
        else:
            lineToParse += "000" + tab + "000" + tab  + "000" + "\n"

else:
    continue
```

```
            print (lineToParse,end="")
            self.outputFile.write(lineToParse)
        self.outputFile.close()
        self.literalTableFile.close()
        self.symboltableFile.close()

obj = pass2()
obj.parseFile()
```

OUTPUT:

```
Literal Table:
0 211
1 212
2 219
Symbol Table:
0 217
1 202
2 218
3 214
4 202
5 216
Machine Code:
LC OPCODE OP1 OP2
200 004 001 211
201 005 001 217
202 004 001 217
203 004 003 218
204 001 003 212
205 004 001 217
206 004 003 218
207 004 001 217
```

```
208 004 003 218

209 004 001 217

210 007 006 214

211 000 000 005

212 000 000 001

213 004 001 217

214 002 001 219

215 007 001 202

216 000 000 000

204 003 003 218

217 000 000 000

218 000 000 000

('C', 1) 000 000 001
```