import unittest

from TestUtils import TestLexer

class LexerSuite(unittest.TestCase):

    """test ID"""

    def test1(self):

        testcase = " aBc a b2 \_c1 \_1 "

        expect = "aBc,a,b2,\_c1,\_1,<EOF>"

        self.assertTrue(TestLexer.test(testcase, expect, 101))

    """test INT\_TYPE"""

    def test2(self):

        testcase = "1 111 11\_11\_01  123\_45\_6"

        expect = "1,111,111101,123456,<EOF>"

        self.assertTrue(TestLexer.test(testcase, expect, 102))

    """test FLOAT\_TYPE"""

    def test3(self):

        testcase = "2015043.E10 .E+2 1.3 20\_1.5043 "

        expect = "2015043.E10,.E+2,1.3,201.5043,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,103))

    """test STRING\_TYPE"""

    def test4(self):

        testcase = """ "abc123\\n" "ABC?\_\\t321" "My id is 2015043" "\\\\ \\t \\f \\n \\r \\" \\'"  """

        expect = """abc123\\n,ABC?\_\\t321,My id is 2015043,\\\\ \\t \\f \\n \\r \\" \\',<EOF>"""

        self.assertTrue(TestLexer.test(testcase,expect,104))

    """test COMMENT"""

    def test5(self):

        testcase = "//Hello My name is Vien. How are you?"

        expect = "<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,105))

    """test multiple lines comment """

    def test6(self):

        testcase = "/\* Hello \n My name is \n Vien \n How \n are \n you \*/"

        expect = "<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,106))

    """test seperators"""

    def test7(self):

        testcase = "(){}[].,;:="

        expect = "(,),{,},[,],.,,,;,:,=,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,107))

    """test operators"""

    def test8(self):

        testcase = "+ - \* / % ! && || == != > >= < <= ::"

        expect = "+,-,\*,/,%,!,&&,||,==,!=,>,>=,<,<=,::,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,108))

    """test keywords"""

    def test9(self):

        testcase = """auto break boolean do else false

                    float for function if integer return

                    string true while void out continue of inherit array"""

        expect = "auto,break,boolean,do,else,false,float,for,function,if,integer,return,string,true,while,void,out,continue,of,inherit,array,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,109))

    """test ERROR\_TOKEN"""

    def test10(self):

        testcase = "? My student ID is 2015043"

        expect = "Error Token ?"

        self.assertTrue(TestLexer.test(testcase,expect,110))

    def test11(self):

        testcase = "vien.nguyen999@hcmut.edu.vn"

        expect = "vien,.,nguyen999,Error Token @"

        self.assertTrue(TestLexer.test(testcase,expect,111))

    def test12(self):

        testcase = "500\_0$)("

        expect = "5000,Error Token $"

        self.assertTrue(TestLexer.test(testcase,expect,112))

    def test13(self):

        testcase = "&&&"

        expect = "&&,Error Token &"

        self.assertTrue(TestLexer.test(testcase,expect,113))

    def test14(self):

        testcase = "2^3"

        expect = "2,Error Token ^"

        self.assertTrue(TestLexer.test(testcase,expect,114))

    def test15(self):

        testcase = "||testbtl|"

        expect = "||,testbtl,Error Token |"

        self.assertTrue(TestLexer.test(testcase,expect,115))

    """test unclose string"""

    def test16(self):

        testcase = """ "my name is vien """

        expect = "Unclosed String: my name is vien "

        self.assertTrue(TestLexer.test(testcase,expect,116))

    def test17(self):

        testcase = """ "\nabc """

        expect = "Unclosed String: "

        self.assertTrue(TestLexer.test(testcase,expect,117))

    def test18(self):

        testcase = """ "my name\n is vien" """

        expect = "Unclosed String: my name"

        self.assertTrue(TestLexer.test(testcase,expect,118))

    def test19(self):

        testcase = """ "my name\r is vien" """

        expect = "Unclosed String: my name"

        self.assertTrue(TestLexer.test(testcase,expect,119))

    def test20(self):

        testcase = """ "2015043\n\\v """

        expect = "Unclosed String: 2015043"

        self.assertTrue(TestLexer.test(testcase,expect,120))

    def test21(self):

        testcase = """ "vien\n\r\\v" """

        expect = "Unclosed String: vien"

        self.assertTrue(TestLexer.test(testcase,expect,121))

    def test22(self):

        testcase = """ "\\"\\n\\t\n\\f\\b" """

        expect = 'Unclosed String: \\"\\n\\t'

        self.assertTrue(TestLexer.test(testcase,expect,122))

    def test23(self):

        testcase = """ "abcdef" \n \t "abc\t\tdef\n" """

        expect = "abcdef,Unclosed String: abc\t\tdef"

        self.assertTrue(TestLexer.test(testcase,expect,123))

    def test24(self):

        testcase = """ 2015043 "\rabc" 2015043 """

        expect = "2015043,Unclosed String: "

        self.assertTrue(TestLexer.test(testcase,expect,124))

    def test25(self):

        testcase = """ "\\t\nabc" """

        expect = "Unclosed String: \\t"

        self.assertTrue(TestLexer.test(testcase,expect,125))

    """test illegal escape"""

    def test26(self):

        testcase = """ "my name is\\xvien" """

        expect = "Illegal Escape In String: my name is\\x"

        self.assertTrue(TestLexer.test(testcase,expect,126))

    def test27(self):

        testcase = """ "\\m\nabc" """

        expect = "Illegal Escape In String: \\m"

        self.assertTrue(TestLexer.test(testcase,expect,127))

    def test28(self):

        testcase = """ "\\ " """

        expect = "Illegal Escape In String: \\ "

        self.assertTrue(TestLexer.test(testcase,expect,128))

    """random test"""

    def test29(self):

        testcase = """ trivien "trivien" float """

        expect = "trivien,trivien,float,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,129))

    def test30(self):

        testcase = """ if (x==2) else x + 2 """

        expect = "if,(,x,==,2,),else,x,+,2,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,130))

    def test31(self):

        testcase = """ for \_j\_ and " or """

        expect = "for,\_j\_,and,Unclosed String:  or "

        self.assertTrue(TestLexer.test(testcase,expect,131))

    def test32(self):

        testcase = """ x = y + arr[0,1] """

        expect = "x,=,y,+,arr,[,0,,,1,],<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,132))

    def test33(self):

        testcase = "x:integer = 201504\_3;"

        expect = "x,:,integer,=,2015043,;,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,133))

    def test34(self):

        testcase = """ "a" + .5 """

        expect = "a,+,.,5,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,134))

    def test35(self):

        testcase = """do {x % 2\_0\_1\_5\_0\_4\_3}"""

        expect = "do,{,x,%,2015043,},<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,135))

    def test36(self):

        testcase = """ <EOF> """

        expect = "<,EOF,>,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,136))

    def test37(self):

        testcase = """ \n\t\r\\t """

        expect = "Error Token \\"

        self.assertTrue(TestLexer.test(testcase,expect,137))

    def test38(self):

        testcase = """ main: function integer () """

        expect = "main,:,function,integer,(,),<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,138))

    def test39(self):

        testcase = """ 2015043\n\r\t "" """

        expect = "2015043,,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,139))

    def test40(self):

        testcase = """ "BKHCM"::"CSE K20"""

        expect = "BKHCM,::,Unclosed String: CSE K20"

        self.assertTrue(TestLexer.test(testcase,expect,140))

    def test41(self):

        testcase = """ 20\_e15043 """

        expect = "20,\_e15043,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,141))

    def test42(self):

        testcase = """ 2015043\_tri.vien-14112002 """

        expect = "2015043,\_tri,.,vien,-,14112002,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,142))

    def test43(self):

        testcase = """ .E.E-30 """

        expect = ".,E,.E-30,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,143))

    def test44(self):

        testcase =  "2\_015043.E "

        expect = "2015043.,E,<EOF>"

        self.assertTrue(TestLexer.test(testcase, expect, 144))

    def test45(self):

        testcase = "01411\_2002"

        expect = "0,14112002,<EOF>"

        self.assertTrue(TestLexer.test(testcase, expect, 145))

    def test46(self):

        testcase = "2015043\_"

        expect = "2015043,\_,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,146))

    def test47(self):

        testcase = """ trivien\_"abc\\t"||' """

        expect = "trivien\_,abc\\t,||,Error Token '"

        self.assertTrue(TestLexer.test(testcase,expect,147))

    def test48(self):

        testcase = "//trivienCSE\n \_\_\_"

        expect = "\_\_\_,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,148))

    def test49(self):

        testcase = "/\*abcd\*?dcba\*/ \n int i = 10"

        expect = "int,i,=,10,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,149))

    def test50(self):

        testcase = "7E9\*0e.2"

        expect = "7E9,\*,0,e,.,2,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,150))

    def test51(self):

        testcase = ""

        expect = "<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,151))

    def test52(self):

        testcase = "print(x,y)"

        expect = "print,(,x,,,y,),<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,152))

    def test53(self):

        testcase = """ C++ is hard"?" """

        expect = "C,+,+,is,hard,?,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,153))

    def test54(self):

        testcase = "if this: \\t, then That\_"

        expect = "if,this,:,Error Token \\"

        self.assertTrue(TestLexer.test(testcase,expect,154))

    def test55(self):

        testcase = """ "abc" \_2015043 "$"  """

        expect = "abc,\_2015043,$,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,155))

    def test56(self):

        testcase = "do{1+2}\"abcd "

        expect = "do,{,1,+,2,},Unclosed String: abcd "

        self.assertTrue(TestLexer.test(testcase,expect,156))

    def test57(self):

        testcase = "\t\r\n "

        expect = "<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,157))

    def test58(self):

        testcase = "true2015043||false"

        expect = "true2015043,||,false,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,158))

    def test59(self):

        testcase = """ 123\_.e-123 """

        expect = "123,\_,.e-123,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,159))

    def test60(self):

        testcase = """ 201-.e\_-5043 """

        expect = "201,-,.,e\_,-,5043,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,160))

    def test61(self):

        testcase = """ "my\_name\_is\_TV" """

        expect = "my\_name\_is\_TV,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,161))

    def test62(self):

        testcase = """ /\*/\*/\* trivien \*/"""

        expect = "\*,trivien,\*,/,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,162))

    def test63(self):

        testcase = """ a=c\d """

        expect = "a,=,c,Error Token \\"

        self.assertTrue(TestLexer.test(testcase,expect,163))

    def test64(self):

        testcase = """ It's a good grade"""

        expect = "It,Error Token '"

        self.assertTrue(TestLexer.test(testcase,expect,164))

    def test65(self):

        testcase = "abc^xyz"

        expect = "abc,Error Token ^"

        self.assertTrue(TestLexer.test(testcase,expect,165))

    def test66(self):

        testcase = "for(int x := 1;x<=2;x++)"

        expect = "for,(,int,x,:,=,1,;,x,<=,2,;,x,+,+,),<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,166))

    def test67(self):

        testcase = """ \r abcde\_1\_.411e2002 """

        expect = "abcde\_1\_,.411e2002,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,167))

    def test68(self):

        testcase = """ 2\_0\_1\_5\_0\_\_\_\_\_43 """

        expect = "20150,\_\_\_\_\_43,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,168))

    def test69(self):

        testcase = """ " """

        expect = "Unclosed String:  "

        self.assertTrue(TestLexer.test(testcase,expect,169))

    def test70(self):

        testcase = """ Uoc 01\_0 diem PPL """

        expect = "Uoc,0,10,diem,PPL,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,170))

    def test71(self):

        testcase = """ while(false){x=2;} """

        expect = "while,(,false,),{,x,=,2,;,},<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,171))

    def test72(self):

        testcase = """ main:function void(){} """

        expect = "main,:,function,void,(,),{,},<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,172))

    def test73(self):

        testcase = """ a[1,2]= 2015043 """

        expect = "a,[,1,,,2,],=,2015043,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,173))

    def test74(self):

        testcase = """ "\\t\\r\\n\\f\\b\\'\\"\\\\" """

        expect = """\\t\\r\\n\\f\\b\\'\\"\\\\,<EOF>"""

        self.assertTrue(TestLexer.test(testcase,expect,174))

    def test75(self):

        testcase = """ /\* "//g"\*/ """

        expect = "<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,175))

    def test76(self):

        testcase = """ // "\naaa" """

        expect = "aaa,Unclosed String:  "

        self.assertTrue(TestLexer.test(testcase,expect,176))

    def test77(self):

        testcase = """ "lo ve u" <3 """

        expect = "lo ve u,<,3,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,177))

    def test78(self):

        testcase = """ a\nb\nc\n""\r  """

        expect = "a,b,c,,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,178))

    def test79(self):

        testcase = """ "v""i""e""n"  """

        expect = "v,i,e,n,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,179))

    def test80(self):

        testcase = """ "  "" " " ""  " """

        expect = "  , , ,  ,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,180))

    def test81(self):

        testcase = """  "" "" "" "" """

        expect = ",,,,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,181))

    def test82(self):

        testcase = """ \_1\_E-4.Ea\_11 """

        expect = "\_1\_E,-,4.,Ea\_11,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,182))

    def test83(self):

        testcase = """ char\* x """

        expect = "char,\*,x,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,183))

    def test84(self):

        testcase = """:::x:"\\y" """

        expect = "::,:,x,:,Illegal Escape In String: \\y"

        self.assertTrue(TestLexer.test(testcase,expect,184))

    def test85(self):

        testcase = """ \_2015043 + 1411e + 2002. """

        expect = "\_2015043,+,1411,e,+,2002.,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,185))

    def test86(self):

        testcase = """ 26%2 == !(true) """

        expect = "26,%,2,==,!,(,true,),<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,186))

    def test87(self):

        testcase = """ "{}\\" """

        expect = """Unclosed String: {}\\" """

        self.assertTrue(TestLexer.test(testcase,expect,187))

    def test88(self):

        testcase = """ "201\_5043" 20\_15043 """

        expect = "201\_5043,2015043,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,188))

    def test89(self):

        testcase = """ ;:;,vien """

        expect = ";,:,;,,,vien,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,189))

    def test90(self):

        testcase = """return n\*n/n """

        expect = "return,n,\*,n,/,n,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,190))

    def test97(self):

        testcase = """ HCMC\_HCMUT """

        expect = "HCMC\_HCMUT,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,197))

    def test98(self):

        testcase = """ 9999\_()e10 """

        expect = "9999,\_,(,),e10,<EOF>"

        self.assertTrue(TestLexer.test(testcase,expect,198))

    def test99(self):

        testcase = """ m.%!~&| """

        expect = "m,.,%,!,Error Token ~"

        self.assertTrue(TestLexer.test(testcase,expect,199))

    def test100(self):

        testcase = """ ' """

        expect = "Error Token '"

        self.assertTrue(TestLexer.test(testcase,expect,200))