## Centro Universitário de Belo Horizonte – Uni-BH

## Instituto de Engenharia e Tecnologia – IET

## Curso: Ciência da Computação

## Disciplina: Matemática Discreta e Teoria da Computação

## Professor: Moises Henrique Ramos Pereira

Integrantes: **Mateus Augusto Mendonça Resende,**

**Matheus Gomes Silva,**

**Wilson Queiroz**

## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **Trabalho Prático de Teoria da Computação –**

## **Implementação de um AFD**

Exemplos de entradas:

Corretas:

@Test

**public** **void** testIdWithUnderscore() **throws** Exception {

String expected = "[ID]";

String actual = t.read("$a\_bc");

*assertEquals*(expected, actual);

}

@Test

**public** **void** testIdWithoutUnderscore() **throws** Exception {

String expected = "[ID]";

String actual = t.read("$abc");

*assertEquals*(expected, actual);

}

@Test(expected = CharacterNotMappedException.**class**)

**public** **void** testIdBeginsWithUnderscore() **throws** Exception {

t.read("$\_bc");

}

@Test

**public** **void** testIdEndsWithUnderscore() **throws** Exception {

String expected = "[ID]";

String actual = t.read("$a\_");

*assertEquals*(expected, actual);

}

@Test

**public** **void** testTwoIdsWithUnderscore() **throws** Exception {

String expected = "[ID, ID]";

String actual = t.read("$oiu\_ $p\_bc");

*assertEquals*(expected, actual);

}

@Test

**public** **void** testTwoInts() **throws** Exception {

String expected = "[NumInt, NumInt]";

String actual = t.read("987234526235623462317895672 2");

*assertEquals*(expected, actual);

}

@Test

**public** **void** testSingleInt() **throws** Exception {

String expected = "[NumInt]";

String actual = t.read("34716987654987324698");

*assertEquals*(expected, actual);

}

@Test

**public** **void** testReal() **throws** Exception {

String expected = "[NumReal]";

String actual = t.read("657987,6432");

*assertEquals*(expected, actual);

}

@Test

**public** **void** testReal2() **throws** Exception {

String expected = "[NumReal]";

String actual = t.read("657987,6");

*assertEquals*(expected, actual);

}

@Test

**public** **void** testReal3() **throws** CharacterNotMappedException, StringNotClosedException {

String expected = "[NumReal]";

String actual = t.read("6,6432");

*assertEquals*(expected, actual);

}

@Test

**public** **void** testHexaBeginsWithDigit() **throws** CharacterNotMappedException, StringNotClosedException {

String expected = "[NumHex]";

String actual = t.read("4203C0B092");

*assertEquals*(expected, actual);

}

@Test

**public** **void** testHexaBeginsWithLetter() **throws** CharacterNotMappedException, StringNotClosedException {

String expected = "[NumHex]";

String actual = t.read("A42E030B092");

*assertEquals*(expected, actual);

}

@Test

**public** **void** testHexaSingleLetter() **throws** CharacterNotMappedException, StringNotClosedException {

String expected = "[NumHex]";

String actual = t.read("A");

*assertEquals*(expected, actual);

}

@Test

**public** **void** testEmptyString() **throws** CharacterNotMappedException, StringNotClosedException {

String expected = "[Cadeia]";

String actual = t.read("\"\"");

*assertEquals*(expected, actual);

}

@Test

**public** **void** testStringSingleChar() **throws** CharacterNotMappedException, StringNotClosedException {

String expected = "[Cadeia]";

String actual = t.read("\"a\"");

*assertEquals*(expected, actual);

}

@Test

**public** **void** testStringWithSpecialChars() **throws** CharacterNotMappedException, StringNotClosedException {

String expected = "[Cadeia]";

String actual = t.read("\"a945&504-0@as#df\*fd%&trht(yhj'<>?\"");

*assertEquals*(expected, actual);

}

@Test

**public** **void** testOperator() **throws** CharacterNotMappedException, StringNotClosedException {

String expected = "[Cadeia, Op]";

String actual = t.read("\"2398\"\*");

*assertEquals*(expected, actual);

}

@Test

**public** **void** testMultipleTokens() **throws** CharacterNotMappedException, StringNotClosedException {

String expected = "[Cadeia, Op, NumInt, Op, ID, Op, NumHex]";

String actual = t.read("\"2398\"\*893475 = $abc + B1");

*assertEquals*(expected, actual);

}

@Test

**public** **void** testMultipleStars() **throws** CharacterNotMappedException, StringNotClosedException {

String expected = "[Op, NumReal, Op]";

String actual = t.read("= 5,6\*\*");

*assertEquals*(expected, actual);

}

@Test

**public** **void** testMultipleTokens3() **throws** CharacterNotMappedException, StringNotClosedException {

String expected = "[ID, Op, NumReal, Op, NumHex, Op, NumInt, Op, Cadeia, Op, NumInt]";

String actual = t.read("$valor = 5,6\*\* 1B + 324 + \"Goku\"%1");

*assertEquals*(expected, actual);

}

@Test

**public** **void** testMultipleTokens4() **throws** CharacterNotMappedException, StringNotClosedException {

String expected = "[Op, Cadeia, Op, NumInt]";

String actual = t.read("+ \"Goku\"%1");

*assertEquals*(expected, actual);

}

@Test

**public** **void** testMultipleTokens5() **throws** CharacterNotMappedException, StringNotClosedException {

String expected = "[NumHex]";

String actual = t.read("ABCEDF");

*assertEquals*(expected, actual);

}

Incorretas:

@Test(expected = CharacterNotMappedException.**class**)

**public** **void** testIdWithSingleCharcter() **throws** Exception {

t.read("$");

}

@Test(expected = CharacterNotMappedException.**class**)

**public** **void** testIncorrectCommaReal1() **throws** Exception {

t.read("23546546,");

}

@Test(expected = CharacterNotMappedException.**class**)

**public** **void** testIncorrectCommaReal2() **throws** Exception {

t.read(",128798475");

}

@Test(expected = CharacterNotMappedException.**class**)

**public** **void** testIncorrectCommaReal3() **throws** Exception {

t.read("123543,a");

}

@Test(expected = CharacterNotMappedException.**class**)

**public** **void** testIncorrectCommaReal4() **throws** Exception {

t.read("123543,1234a");

}

@Test(expected = CharacterNotMappedException.**class**)

**public** **void** testWrongHexa() **throws** CharacterNotMappedException, StringNotClosedException {

t.read("A42030B09R2");

}

@Test(expected = StringNotClosedException.**class**)

**public** **void** testStringNotClosed() **throws** CharacterNotMappedException, StringNotClosedException {

t.read("\"a945&504-0@as#df\*fd%&trht(yhj\\'<>?");

}

@Test(expected = StringNotClosedException.**class**)

**public** **void** testEmptyStringNotClosed() **throws** CharacterNotMappedException, StringNotClosedException {

t.read("\"");

}

@Test(expected = CharacterNotMappedException.**class**)

**public** **void** testMultipleTokens2() **throws** CharacterNotMappedException, StringNotClosedException {

t.read("$ A3s");

}

@Test(expected = CharacterNotMappedException.**class**)

**public** **void** testWrongLetterForHexa() **throws** CharacterNotMappedException, StringNotClosedException {

t.read("ABGCEDF");

}

@Test(expected = CharacterNotMappedException.**class**)

**public** **void** testWrongLetterForHexa2() **throws** CharacterNotMappedException, StringNotClosedException {

t.read("abcdef");

}