Teste Prático de Compiladores/LFA (Duração 2h30m)

Nome: _____ Nº Mec.: _____

- Faça login no computador seguindo as instruções do docente.
- No directório *Desktop* vai encontrar um conjunto de ficheiros úteis para o exame.
- Utilize o executável `./run-jar` como complemento na especificação do programa.
  Exemplo: `./run-jar p1.txt`
- Desenvolva a linguagem por forma a que os programas de exemplo da linguagem (`p?.txt`) sejam aceites.
- Pode consultar a documentação das classes `Java` usando o comando `view-javadoc`.
  Exemplo: `view-javadoc ParseTreeProperty`
- Tem à sua disposição os comandos de apoio à programação em `ANTLR4`: `antlr4`, `antlr4-build`, `antlr4-run`, `antlr4-clean`, `antlr4-test`, `antlr4-main`, `antlr4-visitor`, `antlr4-listener`
- Utilize o enunciado como rescunho, e no final **entregue-o com o cabeçalho preenchido**.
- Caso pretenda desistir deve indicar essa decisão no enunciado e executar o comando: `desisto`

Problema: Pretende-se implementar um interpretador para cálculo com fracções inteiras. Como exemplo inicial, considere o seguinte programa:

```
– p1.txt
display 2/3;    – escreve na consola a fracção 2/3
display 4;      – escreve na consola a fracção 4
c <= 1/4;       – guarda a fracção 1/4 na variável c
display c;      – escreve na consola a fracção armazenado na variável c
```

Nota 1: partindo das instruções exemplificadas, tente tornar a linguagem o mais genérica possível.

Nota 2: os identificadores para variáveis contêm apenas letras minúsculas.

Nota 3: considere que as fracções literais (ex: 1/4, 2) envolvem sempre números inteiros sem sinal (podendo não ter denominador).

Nota 4: não se esqueça das verificações semânticas. Existem ficheiros `err?.txt` para o ajudar nesse fim.

a) Implemente em `ANTLR4`, uma gramática `FracLang` para esta linguagem. [4 valores]

b) Implemente um interpretador que faça a verificação semântica e execute as instruções desta linguagem. [4 valores]

c) Altere a gramática e o interpretador por forma a permitir a realização das seguintes operações sobre fracções (ver programa `p2.txt`): [6 valores]

- soma/subtracção/multiplicação/divisão de fracções (operadores $+ \; - \; * \; :$), com as precedências naturais.[1].
- operadores prefixos unários ($+ \; -$): aplicáveis a qualquer expressão. Este operador deve ser prioritário relativamente a todos os anteriores.
- parêntesis: este operador serve para impor prioridades na realização de operações com fracções.

[1] $(a/b) \pm (c/d) = (ad \pm cb)/(bd)$ $\quad (a/b) * (c/d) = (ac)/(bd)$ $\quad \frac{a/b}{c/d} = (a/b) * (d/c)$

```
– p2.txt
display 1/2-4+5+8/4;
n  <=  2;
m <=  1/8;
display n*n*m;
display n+m:m-n;
display -n;
display -m;
o<= 1-1/1;
p<=-(o+n  -  m*5);
display p;
```

d) Faça com que o interpretador leia o programa a partir de um ficheiro (cujo nome é passado como argumento do programa), e altere a gramática por forma a permitir a entrada de fracções pelo utilizador (ver programa `p3.txt`). [3 valores]

```
– p3.txt
display read "x"; – pede uma fracção ao utilizador apresentando o texto "x: "e escreve-a na consola
media <= (read "f1" + read "f2"):2;
display media;
```

e) Acrescente um operador que reduza a fracção à sua forma mais simples (ver programa `p4.txt`). [3 valores]

Uma fracção pode ser reduzida sempre que o numerador e o denominador tiverem divisores em comum (maiores do que 1). Pode utilizar o algoritmo de Euclides para determinar o máximo divisor comum entre números inteiros (mod é o resto da divisão inteira):

$$mdc(a,b) = \begin{cases} a & \text{se } b = 0 \\ mdc(b, a \bmod b) & \text{se } b \neq 0 \end{cases} .$$

Considere as seguinte prioridades entre operadores (por ordem decrescente): operadores unários prefixos, multiplicação/divisão, soma/subtracção, e, por fim, redução de fracções.