Universidade Federal de Viçosa – Campus UFV-Florestal
Ciência da Computação – Compiladores
Professor: Daniel Mendes Barbosa
Trabalho Prático 0 – Data de entrega: ver no PVANet Moodle

Este trabalho é **individual**. Você deverá entregar um único arquivo PDF, contendo os seus arquivos **lex.l** e **lex2.l** (de acordo com a especificação a seguir e formatado no arquivo PDF da melhor forma possível) e um pequeno **relatório** explicando as decisões que você teve que tomar para que os seus analisadores léxicos gerados pelo flex (um lex mais recente) reconhecesse os tokens conforme foi definido aqui e demonstrado pelo exemplo para o arquivo lex.l, além do arquivo lex2.l construído por você. Este relatório também deverá conter **pelo menos um arquivo de entrada feito por você**, e a saída de seu analisador léxico para tal entrada, discutida, ou seja, explicando o porquê de ter funcionando daquela maneira para o seu exemplo (um para cada um dos dois analisadores léxicos feitos). A entrega deverá ser feita através do PVANet Moodle, até a data limite especificada lá. Se você quiser entregar os arquivos originais em separado e outros arquivos que julgar necessários, você também poderá entregar um arquivo .zip, além do PDF do relatório.

Obs.: boa parte da nota do trabalho poderá ser atribuída com base nos arquivos que você fez e nas justificativas do que eles testam. Outra parte importante é a documentação do que foi feito, referenciando os conceitos aprendidos.

Para escrever o seu arquivo **lex.l** e testá-lo, você precisará da ferramenta flex. Se ela não estiver presente em seu sistema operacional Linux (se você não tiver linux, instale nativamente ou então em uma máquina virtual), você provavelmente poderá instalar através da seguinte linha de comando:

```
sudo apt-get install flex
```

Uma vez instalado, primeiramente crie o seu arquivo lex.l baseado nesse modelo:

```
%{

/*codigo colocado aqui aparece no arquivo gerado pelo flex*/

%}

/* This tells flex to read only one input file */
%option noyywrap

/* definicoes regulares */

delim        [ \t\n]
ws           {delim}+
```

```
%%

{ws}          {/*nenhuma acao e nenhum retorno*/}



%%

/*codigo em C. Foi criado o main, mas podem ser criadas outras funcoes aqui.*/

int main(void)
{
    /* Call the lexer, then quit. */
    yylex();
    return 0;
}
```

Após completar esse arquivo com os padrões especificados a seguir, compile com os seguintes comandos:

```
flex lex.l
gcc lex.yy.c
```

Execute

```
./a.out
```

e então digite o texto a ser analisado. Ou então escreva o texto a ser analisado em um arquivo entrada.txt (exemplo de nome de arquivo) e execute:

```
./a.out < entrada.txt
```

Padrões a serem reconhecidos pelo seu analisador léxico:

Espaços em branco, tabulação e quebra de linha devem ser ignorados.
Número inteiro positivo: qualquer sequencia de um ou mais dígitos, precedidos ou não do símbolo +.
Número inteiro negativo: qualquer sequencia de um ou mais dígitos, precedidos do símbolo -.
Número decimal: qualquer sequencia de um ou mais dígitos seguida de um ponto (.) e de outra sequencia de um ou mais dígitos. **Obs**.: o número decimal também pode ser positivo ou negativo.
Placa: três letras maiúsculas seguidas de um hífen (mesmo caractere dos números negativos) e de 4 dígitos.
Palavra: qualquer sequencia de uma ou mais letras maiúsculas ou minúsculas (sem caractere especial ou letras acentuadas).
Telefone: 4 dígitos seguidos de um hífen (mesmo caractere dos números negativos) e de mais 4 dígitos.

Nome próprio: três ou quatro palavras, tendo necessariamente e exatamente um espaço em branco entre cada par dessas palavras. Um espaço ao final não é necessário.

Obs.: veja que dependendo da entrada, mais de uma combinação dos padrões disponíveis é possível. Você deverá escrever os padrões na ordem de prioridade correta. Veja um exemplo de arquivo de entrada e a respectiva saída do analisador léxico a seguir. Essa saída foi gerada pelas ações (código em C entre chaves para cada padrão), uma vez que não necessitamos retornar os tokens. Queremos apenas imprimir neste trabalho. Para imprimir o LEXEMA do token encontrado, basta imprimir a variável `yytext`.

Arquivo de entrada de exemplo:

```
875878 -3355456 abc5464     abc-5464 ABC-5464   453-2345 9486-0847
Daniel Mendes Barbosa 32.345  Palavra Qualquer 3567-3224
Daniel Mendes Barbosa Daniel Mendes Barbosa    Menezes200
```

Saída após a execução do analisador léxico gerado de acordo com a especificação (**seu analisador deverá produzir exatamente essa saída para esse exemplo**):

```
Foi encontrado um numero inteiro positivo. LEXEMA: 875878
Foi encontrado um numero inteiro negativo. LEXEMA: -3355456
Foi encontrado uma palavra. LEXEMA: abc
Foi encontrado um numero inteiro positivo. LEXEMA: 5464
Foi encontrado uma palavra. LEXEMA: abc
Foi encontrado um numero inteiro negativo. LEXEMA: -5464
Foi encontrado uma placa. LEXEMA: ABC-5464
Foi encontrado um numero inteiro positivo. LEXEMA: 453
Foi encontrado um numero inteiro negativo. LEXEMA: -2345
Foi encontrado um telefone. LEXEMA: 9486-0847
Foi encontrado um nome proprio. LEXEMA: Daniel Mendes Barbosa
Foi encontrado um numero com parte decimal. LEXEMA: 32.345
Foi encontrado uma palavra. LEXEMA: Palavra
Foi encontrado uma palavra. LEXEMA: Qualquer
Foi encontrado um telefone. LEXEMA: 3567-3224
Foi encontrado um nome proprio. LEXEMA: Daniel Mendes Barbosa Daniel
Foi encontrado uma palavra. LEXEMA: Mendes
Foi encontrado uma palavra. LEXEMA: Barbosa
Foi encontrado uma palavra. LEXEMA: Menezes
Foi encontrado um numero inteiro positivo. LEXEMA: 200
```

Faça da mesma forma que você fez o arquivo lex.l e da mesma forma que o discutiu no relatório, o arquivo lex2.l.

No arquivo lex2.l você deve fazer um analisador léxico que reconheça outros padrões de lexema, definidos por você, e explicados no relatório. O relatório deverá conter ainda pelo menos um exemplo de entrada para o lex2.l e um exemplo de saída, bem como as explicações que você julgar necessárias para esta execução.