




	Primeira Avaliação	Nota: 5,0
Curso:	Ciência da Computação	
Disciplina:	Compiladores	
Aluno(a):		Data: 18/04/2023

- 1) Considere uma linguagem que permita, por exemplo, "MEDIA1", "MEDIA1A" e "M" como identificadores; mas não permita "1VAR" ou "1". Sendo "L" uma letra e "D" um dígito (número natural), marque a expressão regular que especifica formalmente os identificadores desta linguagem. (0,5 pt)
- a) ☐ () $(L | D)^+$
b) ☐ () $(L | D)^*$
c) ☐ () $L(L | D)^+$
d) ☒ (x) $L(L | D)^*$
e) ☐ () $(L | D) | (L | D)^*$
f) ☐ () $(L | D)^* (L | D)^*$
- 2) Marque a opção que melhor descreve a solução utilizada na implementação do analisador léxico da linguagem Tiny. (0,5 pt)
- a) ☐ () A solução utilizada é a que representa a função programa do autômato como um vetor bidimensional indexado pelos estados e símbolos do alfabeto. A cada leitura de caractere do arquivo de código fonte esta estrutura é consultada para determinar o novo estado do autômato. O estado atual é representado como uma variável global.
- b) ☐ () A solução adotada consiste em utilizar dois tipos de estruturas: "IF" para testar transições entre estados diferentes e "while" para testar transições de um estado para ele mesmo. A informação sobre o estado atual não é representada explicitamente como em uma variável.
- c) ☒ (x) A solução consiste em utilizar um laço para controlar a execução do autômato até que a condição de parada (atingir um estado final) ocorra. São também utilizados um "switch" para controlar as transições entre estados do autômato e uma variável global que representa seu estado corrente. e
- d) ☐ () A solução utilizada é a que representa a função programa do autômato como um vetor bidimensional indexado pelos estados e símbolos do alfabeto. A cada leitura caractere do arquivo de código fonte esta estrutura é consultada para determinar o novo estado do autômato. Caso a função programa não seja definida para algum par de argumentos (estado corrente e caractere lido do arquivo fonte), isto configura uma situação de erro.
- e) ☐ () A solução consiste em utilizar exclusivamente a estrutura do tipo "IF" para testar transições entre estados. Há também um laço mais externo aos "IFs" que verifica as condições de parada do autômato: atingir um estado final e o código fonte todo lido. A informação sobre o estado corrente é representada implicitamente pelo símbolo lido do arquivo fonte.


3) Sobre a função *reservedlookup*, do analisador léxico da linguagem Tiny, marque a opção verdadeira. (0,5 pt)

- a) () Chama a função *getToken* que por sua vez retorna a próxima palavra reservada a partir do arquivo de código fonte a ser compilado.
- b) () É uma função *void*
- c) () Recebe como argumentos o arquivo de código fonte a ser compilado e o vetor de palavras reservadas da linguagem e verifica a localização destas dentro do código fonte.
- d) () Verifica se a variável global *tokenString* contém um lexema válido na linguagem.
- e) () Retorna à função *getToken* o vetor de palavras reservadas da linguagem.
- f) (X) Nenhuma das anteriores. 

4) Sobre o código do Analisador Léxico da Linguagem Tiny marque V (Verdadeiro) ou F (Falso). (1 pt)

- a) (F) O procedimento *reservedLookup* recebe como argumento um arquivo de código fonte e realiza uma busca neste arquivo por palavras reservadas. 
- b) (F) A solução para implementação utilizada no analisador léxico de Tiny é a que representa a função programa do autômato como uma matriz indexada pelos estados e símbolos do alfabeto. 
- c) (F) A função *UngetNextChar()* decrementa de um o campo *linePos* de uma variável do tipo *TokenType*. 
- d) (F) *TokenType* é a variável que armazena o lexema do *token* corrente. 

5) Sobre a função *getToken*, do analisador léxico da linguagem Tiny, marque a opção correta: (0,5 pt)

- a) () Chama a função *getnextchar* que por sua vez reconhece e retorna a primeira sequência de caracteres que casa com o padrão de algum tipo de token da linguagem.
- b) () Recebe como argumento uma árvore sintática e retorna um conjunto de tokens.
- c) () Retorna 1 ou 0, representando respectivamente se uma sequência de caracteres lida é ou não um token da linguagem.
- d) () Verifica se uma sequência de caracteres lido é uma palavra reservada da linguagem.
- e) () É a função que implementa a fita e a cabeça de leitura da fita, mas não a função programa, de um AFD
- f) (X) Nenhuma das opções está correta. 

6) Sobre as Expressões Regulares, qual a sua relevância no contexto da análise léxica? Marque uma das opções. (0,5 pt)

- a) () Permitem especificar todos os tipos de marcas da linguagem exceto as palavras reservadas.
- b) () Permitem especificar todos os tipos de marcas da linguagem exceto os operadores aritméticos e relacionais.
- c) () Sua relevância está no fato de que permitem especificar formalmente todas as marcas de uma linguagem, o que não é possível com um autômato.

- d) () Correspondem a uma especificação em mais alto nível de abstração de uma implementação de um AFD segundo a solução na qual a função programa é implementada como uma matriz indexada por estados e símbolos do alfabeto.
- e) () Corresponde a uma representação gráfica dos caminhos de processamento para a aceitação dos tokens de uma linguagem.
- f) ☒ Nenhuma das opções está correta. *e*

7) Sobre o compilador e as fases do processo de compilação marque V (verdadeiro) ou F (falso): (1 pt)

- a) (**F**) A análise léxica tem como função principal reconhecer os tokens de uma linguagem e construir uma árvore que tenha os tokens como folhas. *e*
- b) (**F**) A análise léxica recebe como entrada uma sequência de tokens e verifica se estes estão concatenados de forma correta. *e*
- c) (**F**) A análise léxica é independente da máquina alvo, mas para construir o analisador sintático é necessário conhecer a arquitetura da máquina que irá executar o código compilado. *e*
- d) (**F**) As fases podem ser divididas em frente e fundo, sendo que a análise léxica e a sintática estão na frente e a análise semântica e geração de código estão no fundo. *e*

8) Com relação ao analisador léxico da linguagem Tiny pergunta-se: Qual das linhas em um arquivo de código fonte Tiny geraria exatamente 2 erros léxicos? Marque uma das opções. (0,5 pt)

- a) () `media1 - 5 +`
- b) () `media++5`
- c) ☒ `media::5` *e*
- d) () `media=:5`
- e) () Nenhuma das opções anteriores