CET087 – Conceitos de Linguagens de Programação – P1 – Segunda feira, dia 18 de abril de 2022

5,5	Matrícula:	Nome:
	20191028	Igor Lima Rocha

0,5

1. **(2,0 pontos)**
   a) ~~Descreva~~ cada um dos paradigmas de linguagens de programação (mínimo 5 linhas cada).

+/-

   b) Ligue a instância da linguagem de programação ao paradigma correspondente

Instancia de semântica	
Java	
Lisp	
Prolog	
C	

	Modelo Semântico
	Imperativo
	Declarativo
	Funcional
	Orientado a objeto

2,0

2. **(2,0 pontos)** Escrever os seguintes programas p-code. Use como referência a implementação C fornecida em folha separada.
   2.1.- Soma dos números 7 e 11

ERRO

   ~~2.2.-~~ Soma dos números de 1 até 100

2,0

3. **(2,0 pontos)** Escrever os seguintes programas p-code. Use como referência a implementação C fornecida em folha separada.
   3.1.- Fibonacci(5) ITERATIVO

ERRO

   ~~2.2.-~~ Fatorial(4) ITERATIVO

1,0

4. **(2,0 pontos)** Considerando a gramática: $p_1 : E \rightarrow 0$, $p_2 : E \rightarrow 1$, $p_3 : E \rightarrow (EAE)$, $p_4 : A \rightarrow +$, $p_5 : A \rightarrow -$, $p_6 : A \rightarrow *$, $p_7 : A \rightarrow /$.
   a. ~~Determine~~ a derivação mais à esquerda da palavra (1*(0+1)).
   b. ~~Construa~~ a árvore de derivação correspondente à derivação mais à esquerda de (4.a).

2,0

5. **(2,0 pontos)** Considerando a gramática: $p_1 : E \rightarrow 0$, $p_2 : E \rightarrow 1$, $p_3 : E \rightarrow (EAE)$, $p_4 : A \rightarrow +$, $p_5 : A \rightarrow -$, $p_6 : A \rightarrow *$, $p_7 : A \rightarrow /$.
   a. Projete um autômato de pilha que reconheça a mesma linguagem gerada por essa gramática.
   b. Simule a execução do autômato de pilha (5.a) para análise da cadeia (1*(0+1)), registrando as produções correspondentes às transições aplicadas quando conveniente.

Igor Lima Rocha 2019102B2

2 -

// (7 + 11)

2.1 -

```
code[0].f = INT; code[0].l = 0; code[0].a = 4;
code[1].f = LIT; code[1].l = 0; code[1].a = 7;
code[2].f = LIT; code[2].l = 0; code[2].a = 11;
code[3].f = OPR; code[3].l = 0; code[3].a = 2;
code[4].f = STO; code[4].l = 0; code[4].a = 3;
code[5].f = OPR; code[5].l = 0; code[5].a = 0;
```

// 1+2...+100

2.2 -

```
code[0].f = INT; code[0].l = 0; code[0].a = 5;
code[1].f = LIT; code[1].l = 0; code[1].a = 0;
code[2].f = STO; code[2].l = 0; code[2].a = 3;
code[3].f = LIT; code[3].l = 0; code[3].a = 1;
code[4].f = LOD; code[4].l = 0; code[4].a = 3;
code[5].f = OPR; code[5].l = 0; code[5].a = 2;
code[6].f = LIT; code[6].l = 0; code[6].a = 100;
code[7].f = OPR; code[7].l = 0; code[7].a = 13;
code[8].f = JPC; code[8].l = 0; code[8].a = 2;
code[4].f = STO; code[4].l = 0; code[4].a = 4;
code[5].f = LOD; code[5].l = 0; code[5].a = 3;
code[6].f = LOD; code[6].l = 0; code[6].a = 4;
code[7].f = OPR; code[7].l = 0; code[7].a = 2;
code[8].f = STO; code[8].l = 0; code[8].a = 3;
code[9].f = LOD; code[9].l = 0; code[9].a = 4;
code[10].f = LIT; code[10].l = 0; code[10].a = 1;
code[11].f = OPR; code[11].l = 0; code[11].a = 2;
code[12].f = OPR; code[12].l = 0; code[12].a = 13;
code[13].f = JPC; code[13].l = 0; code[13].a = 5;
code[14].f = OPR; code[14].l = 0; code[14].a = 0;
```

Esqueceu intencional mente ↓

3 -

3.2 - // fatorial (4)

code [0].f = INT, code [0]. l = 0; code [0].a = [ ];
code [1].f = LIT; code [1].l = 0; code [1]. a = 1;
code [2].f = STO; code [2].l = 0; code [2]. a = 3;
code [3].f = LiT; code [3]. l = 0; cod [3]. a = 1;
code [4].f = STO; code [4].l = 0; code [4].a = 4;
code [5].f = LOD; code [5].l = 0; code [5]. a = 3;
code [6].f = LOD; code [6].l = 0; code [6].a = 4;
code [7].f = OPR; code [7].l = 0; code [7]. a = 4;
code [8].f = STO; code [8].l = 0; code [8].a = 3;
code [9].f = LOD; code [9].l = 0; code [9]. a = 4;
code [10].f = LIT; code [10].l = 0; code [10]. a = 1;
code [11].f = OPR; code [11].l = 0; code [11].a = 2;
code [12].f = LiT; code [12].l = 0; code [12]. a = 4; // parada
code [13].f = OPR; code [13].l = 0; code [13]. a = 13;
code [14].f = JPC; code [14].l = 0; code [14]. a = 5;
code [15].f = OPR; code [15].l = 0; code [15]. a = 0;

3.1 -

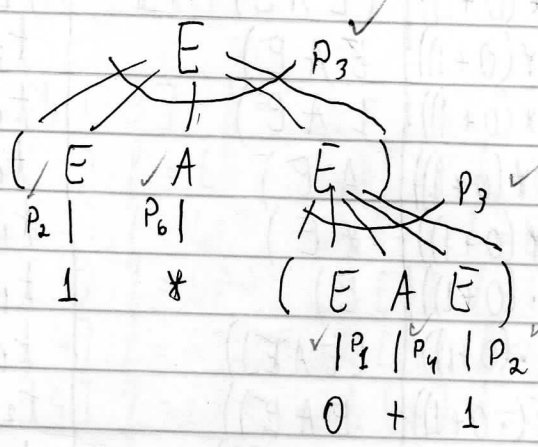
	0; INT; 0; 7;	13; LOD; 0; 4;	1 1
n_1 -	1; LIT; 0; 1;	14; STO; 0; 3;	2
	2; STO; 0; 3;	15; LOD; 0; 6;	3
n_2 -	3; LiT; 0; 1;	16; STO; 0; 4;	5
	4; STO; 0; 4;	17; LOD; 0; 5;	8
i -	5; LIT; 0; 3;	18; LIT; 0; 1;	
	6; STO; 0; 5;	19; OPR; 0; 2;	
result -	7; LiT; 0; 0;	20; STO; 0; 5;	
	8; STO; 0; 6;	21; LOD; 0; 5;	
	9; LOD; 0; 3;	22; LIT; 0; 5; // Parada	
	10; LOD; 0; 4;	23; OPR; 0; 13;	
	11; OPR; 0; 2;	24; JPC; 0; 9;	
	12; STO; 0; 6;	25; OPR; 0; 0;	

DM E : $E \Rightarrow_{P_3} (EAE) \Rightarrow_{P_2} (1AE) \Rightarrow_{P_6} (1*E) \Rightarrow_{P_3} (1*(EAE)) \Rightarrow_{P_1} (1*(1AE))$

4- 4.a- $(1*(0+1)) \Rightarrow_{P_4} (1*(0+E)) \Rightarrow_{P_2} (1*0+1))$

Derivação mais à esquerda é P_2 NÃO!!

4.b-



5- 5.a-

t_i : (read, pop, push)	
t_0 : (E, ε, E)	a numeração das P_i é melhor acompanhar
t_1 : (E, E, (EAE))	
t_2 : (E, E, 0)	
t_3 : (E, E, 1)	— IDEM
t_4 : (A, A, +)	
t_5 : (A, A, -)	
t_6 : (A, A, *)	
t_7 : (A, A, /)	
t_8 : (0, 0, ε)	OK
t_9 : (1, 1, ε)	
t_{10} : (+, +, ε)	
t_{11} : (-, -, ε)	
t_{12} : (*, *, ε)	
t_{13} : (/, /, ε)	
t_{14} : ((, (, ε)	
t_{15} : ( ) ,) , ε)	

5. b -

• w	Stack	t_i
•(1*(0+1))	∅	t_0
•(1*(0+1))	E	t_1
•(1*(0+1))	(EAE)	t_{14}
(•1*(0+1))	EAE)	t_3
(•1*(0+1))	1AE)	t_9 ✓
(1•*(0+1))	AE)	t_6
(1•*(0+1))	*E)	t_{12} ✓
(1*•(0+1))	E)	t_1 ✓
(1*•(0+1))	(EAE))	t_{14}
(1*(•0+1))	EAE))	t_2 ✓
(1*(•0+1))	0AE))	t_8
(1*(0•+1))	AE))	t_4
(1*(0•+1))	+E))	t_{10} ✓
(1*(0+•1))	E))	t_3
(1*(0+•1))	1))	t_9 ✓
(1*(0+1•))	))	t_{15} ✓
(1*(0+1)•)	)	t_{15}
(1*(0+1))•	∅	-

1 -

1. a - O paradigma de uma linguagem funcional indica que a mesma ~~faz~~ recebe instruções, que leva ~~a~~ a máquina à realizar as determinadas tarefas.

1. b -

Lisp — ~~Imperativo~~ FUNCIONAL

C — ~~Funcional~~ IMPERATIVO

Prolog — Declarativo ✓

Java — Orientado a objeto ✓