UNIVERSIDADE FEDERAL DO MARANHÃO		Departamento de Informática - DEINF	1a AVALIAÇÃO	
			P	6,0
Disciplina: Teoria da Computação		Curso: CIÊNCIA DA COMPUTAÇÃO	T	
Código 5607.5	Carga Horária: 60 horas	Créditos: 4.0.0	MEDIA	
Professor: Luciano Reis Coutinho		Email: lrc@deinf.ufma.br		

Primeira Avaliação: Prova Escrita **Data: 15 Maio de 2013.**

Aluno : Gleidson Mendes Costa **Código:** C

INSTRUÇÕES

- A prova deve ser realizada INDIVIDUALMENTE e SEM CONSULTA à livros, anotações, etc. O professor pode ser consultado. No entanto, o papel do professor é tirar dúvidas quanto ao entendimento das questões. O professor não irá atender a pedidos para saber se estão certas ou erradas suas questões. NÃO INSISTAM.
- Cada questão consiste em um enunciado e um conjunto de requisitos que uma resposta aceitável deve satisfazer. Respostas dadas que não atendam aos requisitos podem em última instância ser completamente desconsideradas durante a correção da prova. Tenham sempre em mente os requisitos ao dar as suas respostas.
- A interpretação das questões faz parte da avaliação. Caso ache um enunciado ambíguo ou impreciso escreva na folha de resposta sua interpretação e a correspondente resposta. Todas as questões devem ser interpretadas tendo em vista que foi discutido nas aulas de Teoria da Computação.
- A prova é composta por **7 questões**. Todas as questões devem ser respondidas usando CANETA PRETA ou AZUL. O tempo total de prova é de 100 min.

QUESTÕES

1. **(1,0 ponto)** Considerando as afirmações abaixo:

V I. Um programa pode ser descrito como um conjunto estruturado de instruções que capacitam uma máquina a realizar sucessivamente certas operações básicas e testes sobre os dados iniciais fornecidos como entrada, com o objetivo de transformar estes dados em valores de saída desejáveis.

V II. Um programa monolítico é baseado em desvios condicionais e incondicionais, não possuindo mecanismos explícitos de iteração, subdivisão ou recursão.

V III. Um programa iterativo possui mecanismos de controle de iterações de trechos de programas, e não possui desvios incondicionais.

V IV. Um programa recursivo possui mecanismos de estruturação de subrotinas recursivas, e não possui desvios incondicionais.

V V. Apenas com as noções de programa recursivos, iterativos e monolíticos que foram apresentadas em sala de aula, não é possível definir a noção de computação; necessita-se da noção de máquina.

Assinale a resposta CORRETA:

1,0

(a) Apenas as afirmações I e IV são verdadeiras
(b) Apenas as afirmações I, II e IV são verdadeiras
(c) Apenas a afirmação V é falsa
(d) Apenas a afirmação III é falsa
~~(e)~~ Todas as afirmações são verdadeiras

2. **(1,0 ponto)** Marque a resposta INcorreta: (NÃO RESPONDIDA DE CANETA)

~~(a)~~ Máquinas podem ser definidas como programas em execução, pois cada instrução de qualquer programa sempre tem uma interpretação numa máquina.
(b) Uma computação é, resumidamente, um histórico do funcionamento de uma máquina para um dado programa, considerando um valor inicial. V
(c) Computações podem ser finitas ou infinitas. V
(d) A relação valor de entrada → valor de saída induzida pelas computação de um programa em uma dada máquina dá origem à noção de Função computada. V
(e) De modo geral, funções computadas são funções parciais. V

3. **(1,0 ponto)** Sobre equivalência forte de programas, analise as afirmações abaixo:

F I. Dois programas são fortemente equivalentes se, e somente se, os dois são do mesmo tipo e suas funções computadas são iguais.

F II. Nem todo programa recursivo possui um monolítico fortemente equivalente pelo fato de que o primeiro é mais genérico que o segundo. V

F III. As funções computadas por programas fortemente equivalentes possuem a propriedade de que as mesmas operações podem ser efetuadas em qualquer ordem independente do significado das mesmas, pos a saída é a mesma. F

Marque a alternativa correta:
(a) Apenas a afirmação I é verdadeira;
(b) Apenas a afirmação II é verdadeira; E
(c) As afirmações I e II são falsas;
(d) Apenas a afirmação I é falsa;
(e) Todas as afirmações são falsas.

4. **(1,5 pontos)** Suponha que se escreva M1 $\preccurlyeq$ M2 se a máquina M2 simula a máquina M1, sendo que M1 pode ter em sua definição outros teste e operações a mais do que M2. **Pergunta-se:** Qual a relação entre o poder computacional da máquina M1 em relação a M2, se M1 $\preccurlyeq$ M2. Explique interpretando a idéia de poder computacional em termos de programas P é P', e de funções computadas <P,M1> e <P',M2>. Sua resposta DEVE ter no mínimo 10 linhas de texto.

RESPOSTA:

Se a máquina M_2 é compatível com a máquina M_1 mas podendo possuir mais operações e testes, é de fato coerente afirmar que se determinado programa P que realize uma computação finita em M_1, logo o mesmo irá possuir a mesma computação em M_2, visto que a Máquina M_2 é possui mais operações e testes que a Máquina M_1, mas o inverso nem sempre é válido, uma vez que determinado programa P' pode realizar uma computação finita em M_2, mas não pode realizar a mesma em M_1, podendo gerar uma computação M_2 infinita na mesma.

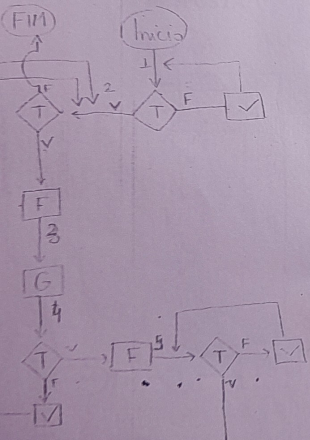
O fato da Máquina M_2 possuir mais operações e testes ajuda a facilitar tal programação, assim determinando que nem todos programas que rodem em M_2 irá funcionar em M_1

5. **(2,0 pontos)** Traduza o programa iterativo P1 abaixo para um programa recursivo equivalente fortemente.

P1:

```
até T
faça (√ );
enquato T
faça (F; G; (se T
                 então F; até T faça (√ )
                 senão √  )
)
```

RESPOSTA:



Monolítico

1: se T vp 2 senão vp 1

2: se T vp 3 senão vp 50

3: faça F vp 4

4: faça G vp 5

5: se T vp 6 senão vp 2

6: faça F vp 7

7: se T vp 2 senão vp 7.

Recursivo

P_1 é R_1

R_1 def (se T então R_2 senão R_1)

R_2 def (se T então R_3 senão R_{50})

R_3 def (F; R_4)

R_4 def (G; R_5)

R_5 def (se T então R_6 senão R_2)

R_6 def (F; R_7)

R_7 def (se T então R_2 senão R_7)

R_{50} def (√)

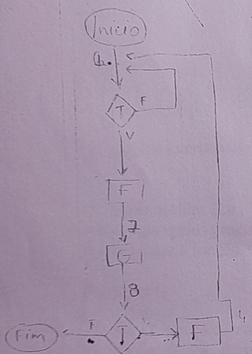
2,0

6. **(2,0 pontos)** Utilizando o método discutido em sala de aula, verifique se os programas P1 (da questão 6) e P2 (abaixo) são ou não são equivalentes fortemente. Lembrete do método: (1) transforme os programas para instruções rotuladas compostas; (2) identifique e simplificando ciclos infinitos (cadeias de conjuntos A's); (3) construa a cadeia de conjuntos B0, B1, ..., Bk de rótulos equivalentes fortemente; (4) caso Bk = {} os programas são equivalentes fortemente, caso contrário, não o são.

P2:

```
1: se T então vapara 2 senão vapara 1
2: faça F vapara 3
3: faça G vapara 4
4: se T então vapara 5 senão vapara 6
5: faça F vapara 1
```

RESPOSTA:



P1

ciclo

1: (V,2)(~~V,1~~)
2: (F,3)(Fim, ε)
3: (G,4)(G,4)
4: (F,5)(V,2)
5: (V,2)(V,5)

A0 {ε}
A1 {2, ε}
A2 {1,2,4,5, ε}
A3 {1,2,3,4,5, ε}

P2

6 : (F,7)(V,6)
7 : (G,8)(G,8)
8 : (F,9)(Fim, ε)
9 : (F7)(V,6)

A0 {ε}
A1 {8, ε}
A2 {7,8, ε}
A3 {6,7,8, ε}

B0 {(,)}

Não equivalentes

ε

0,5

7. **(1,5 pontos)** Considere a máquina `um_reg` definida abaixo:

um_reg = < N, N, N, id , id , {ad, sub}, {zero} >

sendo

id : N → N , tal que id(n)=n

ad: N → N , tal que ad(n)=n+1

sub: N → N, tal que sub(n)=n-1, se n ≠ 0; sub(n)=0, se n=0

zero: N → {verdadeiro, falso}, tal que zero(0)=verdadeiro e zero(n)=falso, se n ≠ 0.

Na máquina um_reg, o programa P :

```
1: se T então vá_para 2 senão vá_para 3
2: faça F vá_para 9
3: faça G vá_para 4
4: se T então vá_para 9 senão vá_para 5
5: faça G vá_para 1
```

computa a função <P, um_reg> : N → N, tal que para qualquer n ∈ N:

<P, um_reg> (n) = 1, se n é par

<P, um_reg> (n) = 0, se n é ímpar

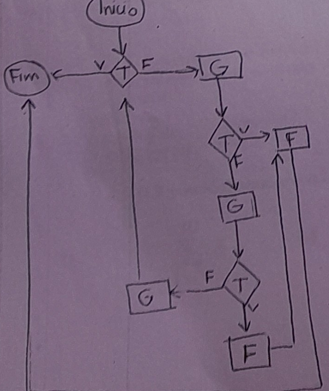
quando T é interpretado como o teste zero, F a operação ad e G a operação sub.

De forma similar (i.e., T sendo o teste zero, F a operação ad e G a operação sub), diga qual a função computada pelo programa Q abaixo na máquina um_reg, escrevendo uma expressão que defina precisamente a função <Q, um_reg> .

Programa Q:

```
1: se T então vá_para 9 senão vá_para 2
2: faça G vá_para 3
3: se T então vá_para 4 senão vá_para 5
4: faça F vá_para 9
5: faça G vá_para 6
6: se T então vá_para 7 senão vá_para 8
7: faça F vá_para 4
8: faça G vá_para 1
```

RESPOSTA:



Este programa executa o processo de encontrar o resto de uma divisão por 3.

<Q, um_reg> : N → N

<Q, um_reg> (n) = 0, se numero é ~~divisível~~ múltiplo por 3. Resto 0.

<Q, um_reg> (n) = 1, número não é ~~divisível~~ múltiplo por 3 e Resto é 1.

<Q, um_reg> (n) = 2, número não é ~~divisível~~ múltiplo por 3 e Resto é 2.

Onde Resto é o resto da divisão de n por 3.