Curso de Formação de Oficiais
Conhecimentos Específicos

ENGENHARIA DE COMPUTAÇÃO

CADERNO DE QUESTÕES

2019

1ª QUESTÃO	Valor: 1,0

A pilha de protocolos TCP/IP *(Transmission Control Protocol / Internet Protocol)*, diferentemente do modelo de referência OSI (*Open Systems Interconnection*), não tem a camada de sessão. No entanto, um protocolo da camada de aplicação pode incluir algumas funcionalidades definidas nessa camada, caso necessário. O protocolo de transferência de arquivos (FTP - *File Transfer Protocol*) faz parte da camada de aplicação da arquitetura TCP/IP e incorpora algumas funções da camada de sessão do modelo OSI. Baseado na afirmativa acima, descreva uma função da camada de sessão que é utilizada pelo protocolo FTP.

2ª QUESTÃO	Valor: 1,0

Você foi contratado para realizar a atualização em um sistema legado de controle de estacionamento. Recentemente, o padrão de placas de veículos no Brasil, utilizando <u>três letras e quatro números</u>, foi atualizado com a inclusão de novas placas no formato <u>três letras, um número, uma letra e dois números</u>. A declaração da classe implementada no sistema em C++, que realiza o armazenamento das informações de um carro e a validação da placa, encontra-se listada abaixo.

```
class Veiculo{
   int ano;                          //atributo que armazena o ano do veículo
   string modelo, marca;             //atributos que armazenam o modelo e a marca do veículo
protected:
   string placa;                     //atributo que armazena a placa do veículo
public:
   Veiculo(int ano, string           //construtor da classe Veiculo que armazena os atributos ano,
modelo, string marca);               //marca e modelo

   virtual string getPlaca();        //método de acesso que retorna a placa do veículo
   virtual bool setPlaca             //método de acesso que valida e armazena a placa do veículo
(string placa);                      //retornando true ou false em função da validação da placa
};
```

Apresente a definição completa da classe derivada, incluindo a implementação dos novos métodos, que atenda às alterações no padrão de placas, mantendo todas as funcionalidades anteriores.

3ª QUESTÃO **Valor: 1,0**

Considere um jogo no qual existem dois jogadores X e Y e o estado do jogo é representado por um inteiro N.

Na sua vez de jogar, X pode escolher entre duas jogadas:

Jogada A, na qual N se torna N + (N mod 13) – 6

Jogada B, na qual N se torna N + (N mod 5) - 2

Na sua vez de jogar, Y pode escolher entre duas jogadas:

Jogada C, na qual N se torna N + 2 *(N mod 9) - 8

Jogada D, na qual N se torna N + ((N mod 6) - 2) * (N mod 8)

No início, N = 50, é a vez de X jogar e seu objetivo é que N seja maior que 50 no final do jogo (o objetivo de Y é o oposto). O jogo acontece durante 4 rodadas: X joga, Y joga, X joga, Y joga e o jogo termina.

Mostre a árvore completa gerada pelo algoritmo minimax, apontando a estratégia (próxima jogada) a ser adotada por X, gerando os sucessores com a seguinte ordem de jogadas da esquerda para a direita: A, B, C, D.

4ª QUESTÃO **Valor: 1,0**

Dado um sistema operacional que utiliza Paginação sob Demanda como gerência da memória e oferece multiprogramação através de um algoritmo de escalonamento, responda às questões abaixo:

a) Considere que o endereço virtual é dividido conforme apresentado na figura abaixo.

Número da página	Deslocamento
20 bits	12 bits

Quantas entradas tem a tabela de páginas e qual o tamanho da página?

b) Considere que um processo tem quatro quadros alocados preenchidos com as páginas indicadas na tabela abaixo. A tabela também contém o instante de carga da página na memória, o instante da última referência, e o valor do bit de referência (R) (os tempos estão em *ticks* de relógio).

Página	Carga	Última ref.	R
0	120	300	1
1	200	250	0
2	140	270	0
3	110	285	1

Dado que o processo deseja acessar a página 4, qual página será substituída pelo algoritmo segunda chance (relógio) e pelo algoritmo LRU (*Least Recently Used*)?

c) Considerando que a maioria das aplicações executadas neste sistema são iterativas, qual é o melhor algoritmo de escalonamento a se usar, o *Round Robin* (Circular) ou o FCFS (*First Come First Served*)?

Justifique sua resposta.

5ª QUESTÃO | **Valor: 1,0**

Considere o alfabeto $\boldsymbol{\Sigma} = \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}\}$. Seja a linguagem L_1 o conjunto de sentenças em $\boldsymbol{\Sigma}^*$ que terminam com a, e nas quais o símbolo a não apareceu antes na sentença. Por exemplo, $ba \in L_1$, mas $aa \notin L_1$. Seja a linguagem L_2 o conjunto de sentenças em $\boldsymbol{\Sigma}^*$ nas quais o último símbolo da sentença não apareceu em uma posição anterior na sentença. Por exemplo $ab \in L_2$, pois b não ocorre em uma posição anterior, mas $aba \notin L_2$.

Em relação ao alfabeto $\boldsymbol{\Sigma}$ e às linguagens L_1 e L_2, escreva as expressões regulares:

a) E_1, que denote a linguagem L_1.

b) E_2, que denote a linguagem L_2.

6ª QUESTÃO | **Valor: 1,0**

Considere um texto constituído por um conjunto de símbolos (ou caracteres) $S = \{s_1,..,s_n\}$, $n>1$.

O problema de codificar um texto de forma a obter uma compactação que seja ótima dentro de certos critérios é definido da seguinte forma:

Para cada símbolo s_i , é conhecida a frequência f_i no texto, para $1 \le i \le n$. Deseja-se atribuir um código binário a cada símbolo de modo a compactar o texto, de forma que nenhum código seja prefixo de outro.

Para os dados de entrada $n = 4$, e para as frequências dos símbolos seguintes,

s_i	A	B	C	D
f_i	5	1	2	1

a) mostre a árvore *T* de prefixo mínima obtida usando o algoritmo de *Huffman* para esses dados de entrada;

b) preencha (no caderno de soluções) a tabela a seguir com as palavras-código construídas a partir da árvore *T*;

s_i	A	B	C	D
código				

c) determine o custo *C(T)* da árvore obtida pelo algoritmo de *Huffman*.

7ª QUESTÃO | **Valor: 1,0**

Dadas as sentenças da lógica de 1ª ordem:

I) $\exists x\alpha(x) \rightarrow \forall x\alpha(x)$

II) $\forall x(\varphi(x) \vee \psi(x)) \rightarrow (\forall x\varphi(x) \vee \forall x\, \psi(x))$

III) $\forall x\exists y\delta(x,y) \vee \exists x\forall y\neg\delta(x,y)$

IV) $\forall x\sigma(x) \rightarrow \neg\forall x\sigma(x)$

a) Informe se cada sentença é uma tautologia, satisfazível mas não tautologia, ou insatisfazível.

b) Para cada sentença, faça o que se pede a seguir:

- Se a sentença for uma tautologia: demonstre esse fato usando a definição de consequência semântica, sem usar equivalências entre sentenças;
- Se a sentença for satisfazível, mas não tautologia: exiba uma interpretação em uma estrutura que a torne verdadeira e em outra que a torne falsa;
- Se a sentença for insatisfazível: demonstre esse fato usando a definição de consequência semântica, sem usar equivalências entre sentenças.

8ª QUESTÃO | **Valor: 1,0**

O esquema relacional abaixo corresponde a uma versão simplificada e adaptada da descrição do banco de dados do Sistema Eletrônico de Votação da Câmara dos Deputados.

Parlamentar(id, nome)

Partido(id, nome, sigla)

MandatoParlamentar(id, data_inicio, data_termino, id_parlamentar, id_partido, uf)

- id_parlamentar referencia Parlamentar(id)
- id_partido referencia Partido(id)

Sessao(id, numero, data)

Votacao(id_sessao, id, proposta_descricao)

- id_sessao referencia Sessao(id)

Participa(id, id_mandato, id_sessao, id_votacao, voto)

- id_mandato referencia MandatoParlamentar(id)
- (id_sessao, id_votacao) referencia Votacao(id_sessao, id)
- Dominio(voto)={‘S’, ‘N’, ‘A’}, onde ‘S’ = Sim; ‘N’ = Não; ‘A’ = Abstenção

Com base no esquema acima, elabore os comandos em SQL que atendam aos seguintes pedidos:

a) Obter as siglas dos partidos com mais de dez parlamentares eleitos para o mandato em vigor.

b) Recuperar os nomes dos parlamentares que participaram, sem se abster, de todas as votações ocorridas durante algum de seus mandatos.

Obs.:

1. Caso opte por usar subconsultas, essas somente poderão constar na cláusula WHERE de suas respostas.
2. Utilize a função *sysdate()* para recuperar a data corrente do sistema.

9ª QUESTÃO | **Valor: 1,0**

Considerando uma linguagem que contenha os lexemas abaixo, apresente o autômato finito que represente os estados do Analisador Léxico.

';' , '+' , '+=' , '++' , ':=' , '!' , '!=' , (a..zA..Z)+ , (0..9)+['.' (0..9)+]

Notação:

( ) → Agrupamento

[ ] → Opcional

\+ → Uma ou mais ocorrências

10ª QUESTÃO | **Valor: 1,0**

Uma Arquitetura de Processador teórica possui 32 registradores (de 32 bits cada), e até 64 possíveis instruções, de 32 bits cada (cada instrução é indicada pelo seu OPCode único).

Cada palavra ocupa 4 Bytes de memória. Existem dois tipos de instrução:

1. R-R, que realiza uma operação entre dois registradores (endereçados por Rs, Rt) , e grava o resultado em um terceiro registrador (endereçado por Rd).

Exemplo:

ADD $s1, $s2, $s3 grava no registrador s1 (Rd) a soma dos valores presentes nos registradores s2 e s3 (Rs, Rt)

2. R-M, que realiza uma operação entre um registrador (endereçado por Rs) e um OFFSET, e pode utilizar outro registrador (endereçado por Rd).

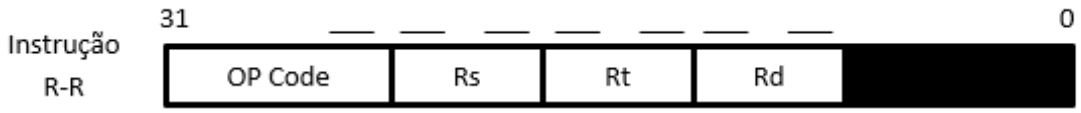
Exemplos:

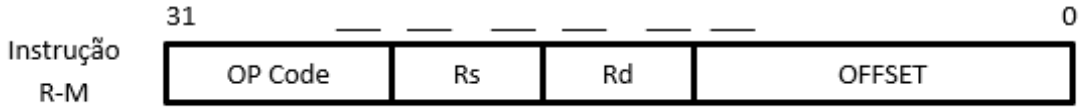
SW $s1, $s2, OFFSET grava no endereço de memória [valor de s2+OFFSET] o valor existente em s1 (Rd);

LW $s1, $s2, OFFSET salva no registrador s1(Rd) o valor presente no endereço de memória [valor de s2+OFFSET] .

Com relação à Arquitetura apresentada,

a) apresente um esquema das instruções R-R e R-M de acordo com o formato abaixo, preenchendo as lacunas (no caderno de solução) com o valor correto da numeração de bits para que seja possível implementar a arquitetura proposta, com OFFSET de tamanho máximo;





b) faça um esquema de processador que implemente a arquitetura proposta, usando os componentes listados abaixo e explicitando as conexões entre eles.

Obs.:

1. Em caso de divisão de trilha, os bits que seguem por cada trilha devem ser identificados.
2. Adicione multiplexadores 2x1 quando necessário.
3. O processador deve executar minimamente as operações R-M: Load word (LW) e Store Word (SW) e operações R-R (como por exemplo ADD).

Componentes:

- Contador de Programa (PC);
- Incrementador de 4 posições;
- Memória de Instruções;
- Banco de Registradores;
- Extensor de sinal para 32 bits;
- Unidade Lógica e Aritmética (ULA) de 32 bits; e
- Memória de dados.