MC626 - Análise e Projeto de Sistemas de Informação

Prova Escrita - 08/05/2014

Resolva as questões abaixo, acumulando seus pontos. Nota 10,0 corresponde a 65 pontos. O total da prova é de 87 pontos.

Questão 1 (24 pontos) Dado o seguinte algoritmo:

```
01. int main(){
02.    int a;
03.    printf("Digite um numero inteiro:");
04.    scanf("%d", &a);
05.    if( ( a % 2 == 0) && (a<100) )
06.       printf("O numero é par e menor que 100\n");
07.    if( ( a % 2 == 0) && (a>=100) )
08.       printf("O numero é par e maior ou igual a 100\n");
09.    if( ( a % 2 != 0) && (a<100) )
10.       printf("O numero é impar e menor que 100\n");
11.    if( ( a % 2 != 0) && (a>=100) )
12.       printf("O numero é impar e maior que 100\n");
13. }
```

a) Desenhe o grafo de fluxo de controle.
b) Crie um conjunto de testes que satisfaça ao critério de todas as instruções.
c) Crie um conjunto de testes que satisfaça ao critério de todos os ramos.

Questão 2 (10 pontos) Considere uma função com duas variáveis de entrada (Cliente e Qtd) e uma variável de saída (Desconto). Cliente pode ser do tipo A, B ou C e Qtd pode variar de 1 a 1000. A função calcula Desconto de acordo com as seguintes regras:

- Clientes do tipo A não recebem desconto se Qtd for inferior a 10; recebem 5% desconto para Qtd 10 e 99; recebem 10% de desconto se Qtd for 100 ou maior.
- Clientes do tipo B recebem 5% de desconto para Qtd abaixo de 10; 10% de desconto para Qtd entre 10 e 99; 20% de desconto se Qtd for 100 ou maior.
- Clientes do tipo C não recebem desconto se Qtd for inferior a 10; 20% de desconto se Qtd estiver entre 10 e 99; 30% de desconto se Qtd for 100 ou maior.

Crie uma tabela de decisão para orientar os casos de teste desta função com base nas regras acima.

Questão 3 (20 pontos) Desenhe o gráfico de fluxo de dados com os conjuntos def(i), c-uso(i) e p-uso(i, j) do seguinte algoritmo:

```
Entradas: tabela, item, chave
Saídas: achou, onde

01. comeco := 1;
02. fim := Tamanho_tabela;
03. achou := falso;
04. while comeco <= fim and not achou do
05.   meio := (comeco + fim) / 2;
06.   if chave > tabela [meio] then
07.      comeco := meio + 1
08.   else if chave = tabela [meio] then
09.           achou := verdade;
10.           onde := meio
11.         else fim := meio - 1
12.   endif;
13. end while;
```

Questão 4 (6 pontos) Um banco usa o seguinte esquema de segurança para permitir o acesso dos usuários às suas contas via internet. Na primeira tela, o usuário se identifica com seu login. Na segunda tela, o usuário fornece a senha. Na terceira tela, o usuário fornece um código gerado por um dispositivo de hardware fornecido pelo banco a cada cliente. Se o login não for válido, o sistema volta à tela inicial. Se o usuário errar a senha por três vezes consecutivas, o login é bloqueado. Se o código for recusado por duas vezes consecutivas, o login é bloqueado.

Crie uma máquina finita de estado que represente esta funcionalidade.

Questão 5 (4 pontos) A informação sobre quem vai realizar os testes está em qual documento: Plano de Testes ou Critério de Testes? E a informação sobre quais métricas serão colhidas durante os testes?

Questão 6 (4 pontos) Acerca do desenvolvimento de um sistema:

a) Defina qualidade de software.

b) Cite e descreva pelo menos 5 (cinco) fatores de qualidade de software.

Questão 7 (6 pontos) Considere o código a seguir:

```
// Soma elementos nas posicoes pares de um vetor de inteiros
// Obtem menor elemento em posicao impar no vetor
01. public class Exemplo {
02. public static void main(String args[]) {
03. int vet[] = {2, 5, 1, 8, 4, 9, 3, 7, 6, 8};
04. int i;
05. int somapar = vet[0];
06. int menorimpar = vet[1];
07. for (i=1; i<vet.length; i++) {
08.  if (i % 2 == 0){
09.    somapar += vet[i]; }
10. else {
11.  if (vet[i] < menorimpar) {
12.  menorimpar = vet[i]; }}}
13. System.out.println("Soma dos pares = " + somapar);
14. System.out.println("Menor nr. impar = " + menorimpar);
```

1. Qual a resposta esperada para o vetor fornecido como entrada de teste?
2. Suponha que foi criado um mutante substituindo, na linha 07, `i=1` por `i=0`. A entrada de teste fornecida mataria esse mutante? Por que?
3. Crie uma entrada diferente da fornecida que mate esse mutante.
4. Crie uma entrada diferente da fornecida que não mate esse mutante.

Questão 8 (2 pontos) Associe itens da primeira lista com os da segunda:

1- erro	( ) comportamento incorreto do software
2- defeito	( ) linha de código incorreta
3- falha	( ) valor calculado de forma incorreta

Questão 9 (2 pontos) Os testes são realizados em várias fases de um desenvolvimento de software. Sabendo que na coluna da esquerda estão listadas diferentes fases de desenvolvimento e na da direita diferentes tipos de testes, numere a coluna da direita de acordo com a da esquerda:

1- Requisitos	( ) Testes de Unidades
2- Análise	( ) Testes de Integração
3- Arquitetura e Projeto	( ) Testes de Sistemas
4- Codificação	( ) Testes de Aceitação

Questão 10 (2 pontos) Crie o grafo de fluxo de chamadas para o programa dado:

```
procedure p()
begin
    call f()
    call g()
end
```

```
procedure f()
begin
    call g()
    call i()
end
```

```
procedure g()
begin
    // two lines
    // of code
end
```

```
procedure i()
begin
    call j()
end
```

```
procedure h()
begin
    call p()
end
```

```
 procedure j()
 begin
     call h()
 end
```

Questão 11 (2 pontos) Analise e comente a correção da seguinte assertiva: "O aumento na medida de complexidade ciclomática de um programa introduz mudanças significativas no refinamento de uma abordagem do tipo caixa-preta".

Questão 12 (2 pontos) Faça uma comparação entre teste e inspeção de software citando as vantagens e desvantagens de cada um deles.

Questão 13 (2 pontos) Qual a diferença entre validação e verificação?

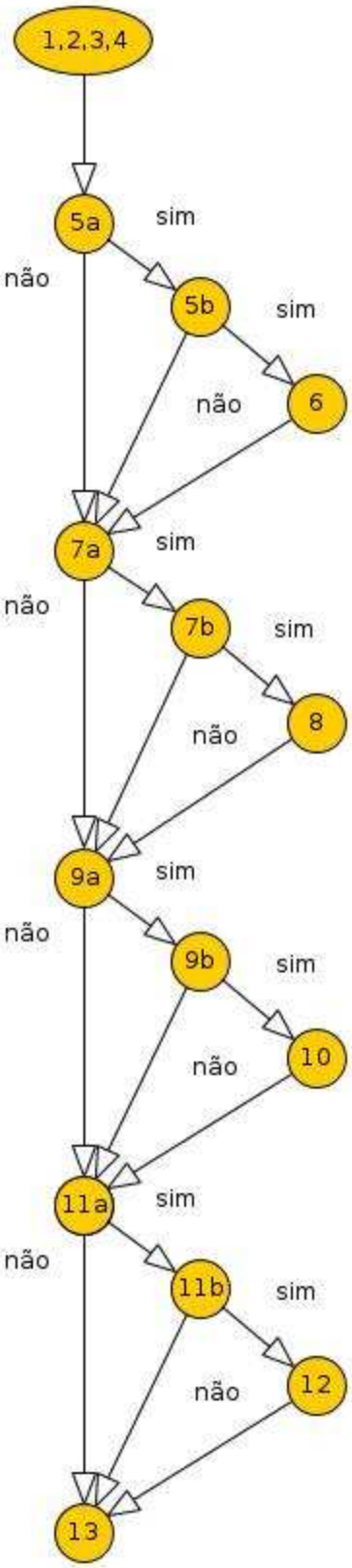
Questão 14 (1 ponto) Classifique as técnicas a seguir como estáticas ou dinâmicas: inspeção, execução simbólica, revisão técnica.

Boa sorte!

MC626 - Prova

Gabarito

Questão 1



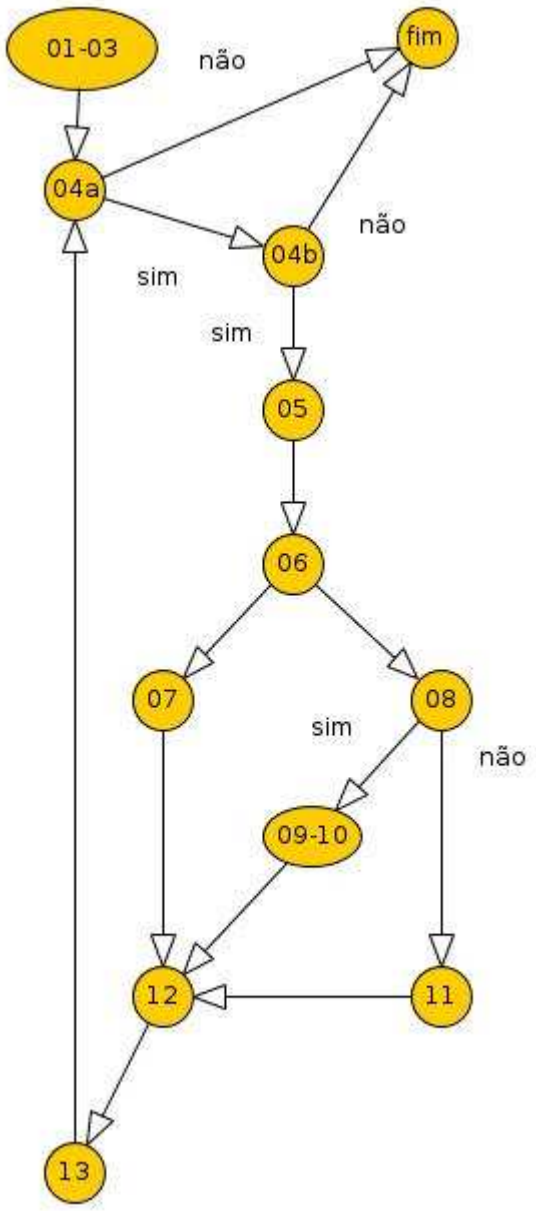
a.
b. Todas as instruções: a=2, a=3, a=102, a=103.
c. Todos os ramos: a=2, a=3, a=102, a=103.

Questão 2

Na seção de "CAUSA", as células vazias contém "F".

CAUSA									
Cliente A	V	V	V						
Cliente B				V	V	V			
Cliente C							V	V	V
Qtde 0-9	V			V			V		
Qtde 10-99		V			V			V	
Qtde 100-1000			V			V			V
EFEITO									
Desc. 0%	X						X		
Desc. 5%		X		X					
Desc. 10%			X		X				
Desc. 20%						X		X	
Desc. 30%									X

Questão 3

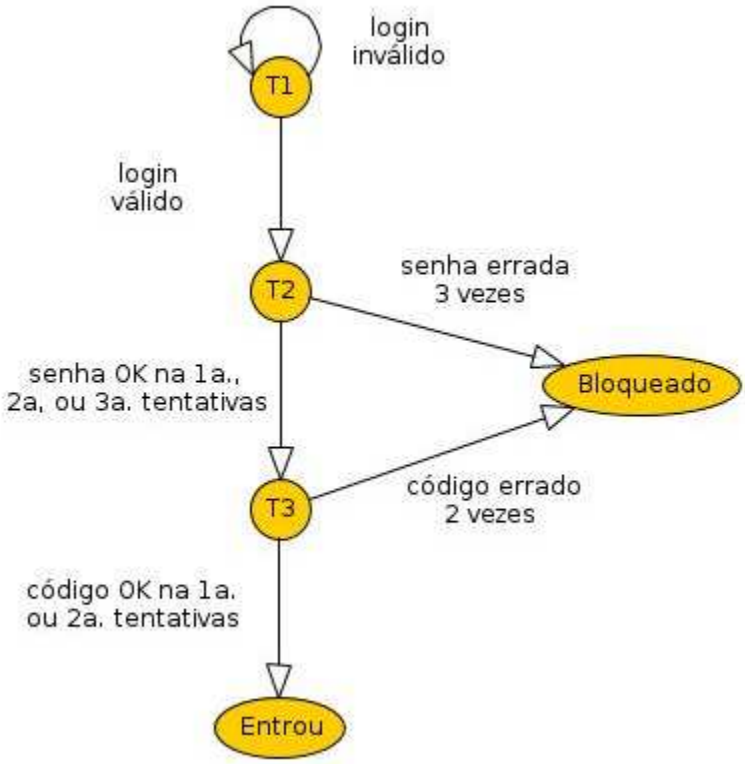


Def(01-03) = {comeco, fim, achou}
Def(05) = {meio}
Def(07) = {comeco}
Def(09-10) = {achou, onde}
Def(11) = {fim}
c-uso(01-03) = {tamanho_tabela}
c-uso(05) = {comeco, fim}
c-uso(07) = {meio}
c-uso(09-10) = {meio}
c-uso(11) = {meio}
p-uso(04a,04b) = {comeco, fim}
p-uso(04a,fim) = {comeco, fim}

p-uso(04b,05) = {achou}
p-uso(04b,fim) = {achou}
p-uso(06,07) = {chave, tabela, meio}
p-uso(06,08) = {chave, tabela, meio }
p-uso(08,09-10) = {chave, tabela, meio}
p-uso(08,11) = {chave, tabela, meio }

Questão 4

Na figura abaixo, T1, T2 e T3 são a primeira tela, segunda tela e terceira tela, respectivamente.



Questão 5

A informação sobre quem vai realizar os testes e quais métricas serão colhidas está no Plano de Testes, pois este documento visa responder às seguintes questões:

- Quem? (equipe de testes; usuários);
- O quê? (requisitos? casos de uso? módulos? ...);
- Quando? (cronograma);
- Onde? (local; ambiente hw e sw);
- Porquê? (critérios de completude);
- Como? (métodos e técnicas).

O Critério de Teste determina um conjunto finito de elementos do modelo de teste que devem ser exercitados durante os testes.

Questão 6

a. Qualidade de software pode ser entendida como o conjunto de características que devem ser alcançadas em um determinado grau para que o produto atenda às necessidades de seus usuários.
b. Fatores de qualidade:
correção: o quanto um programa satisfaz a sua especificação e cumpre os objetivos visados pelo cliente;
confiabilidade: o quanto um programa executa a função pretendida com a precisão exigida;
eficiência: a quantidade de recursos computacionais e de código exigida para que um programa execute sua função;
integridade: o quanto o acesso ao sw ou aos dados por pessoas não autorizadas pode ser controlado;
usabilidade: o quanto de esforço é necessário para aprender, preparar a entrada e interpretar a saída de um programa;
manutenibilidade: o quanto de esforço é necessário para localizar e eliminar erros em um programa;
flexibilidade: o quanto de esforço é necessário para modificar um programa;
testabilidade: o quanto de esforço é necessário para testar um programa a fim de garantir que ele execute a função pretendida;
portabilidade: o quanto de esforço é necessário para transferir um programa de uma plataforma de hw e/ou sw para outra;
reusabilidade: o quanto um programa (ou partes dele) pode ser reutilizado em outros programas;
interoperabilidade: o quanto de esforço é necessário para se acoplar um programa a um outro.

Questão 7

1. Soma dos pares = 16
Menor nr. impar = 5

2. Sim, pois a resposta seria diferente, já que o primeiro elemento (2) seria somado duas vezes aos números pares:
Soma dos pares = 18
Menor nr. impar = 5

3. `int vet[] = { 3, 5, 1, 8, 4, 9, 3, 7, 6, 8 };`

4. `int vet[] = { 0, 5, 3, 8, 4, 9, 3, 7, 6, 8 };`

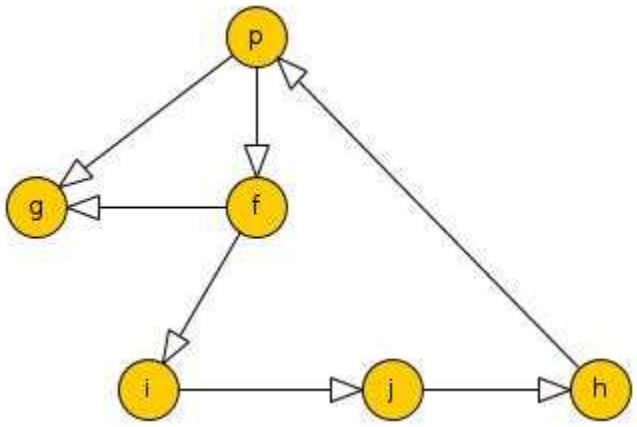
Questão 8

1 - erro (2) comportamento incorreto do software
2 - defeito (3) linha de código incorreta
3 - falha (1) valor calculado de forma incorreta

Questão 9

1 - Requisitos (4) Testes de Unidades
2 - Análise (3) Testes de Integração
3 - Arquitetura e Projeto (2) Testes de Sistemas
4 - Codificação (1) Testes de Aceitação

Questão 10



Questão 11

Complexidade ciclomática diz respeito a um limite máximo para o número de testes necessários para cobrir todas as instruções de um programa. Porém testes caixa-preta (ou funcionais) focam nas entradas e saídas especificadas nos requisitos funcionais e não se preocupam com a estrutura interna (intruções) do programa, pois não têm acesso ao código. O correto seria:

> "O aumento na medida de complexidade ciclomática de um programa introduz mudanças significativas no refinamento de uma abordagem do tipo caixa-branca" .

Questão 12

Inspeções e testes são atividades complementares, e não mutuamente exclusivas.

Inspeções servem para verificar a conformidade com as especificações e se as convenções e padrões de desenvolvimento estão sendo seguidos. Inspeções prescindem da execução do sistema, portanto podem ser aplicadas antes da implementação e ser aplicadas a qualquer representação do sistema: requisitos, projeto, modelo e código, sendo técnicas efetivas para o descobrimento de erros no programa, além de promoverem troca de conhecimento entre os participantes. Por outro lado, inspeções aumentam o custo do processo de desenvolvimento, pois as equipes devem ser bem informadas e ter acesso a especificações precisas, padrões organizacionais devem ser bem definidos e a gerência pode utilizar os achados de inspeção para avaliar indivíduos.

Os testes servem para verificar conformidade com requisitos do usuário e são capazes de verificar requisitos de qualidade (não funcionais). Por outro lado, os testes podem precisar de várias execuções para descobrir certas falhas, pois uma falha pode mascarar outras.

Questão 13

Verificação refere-se ao conjunto de atividades que garantem que o software implementa corretamente as funções especificadas, é o processo de se avaliar um software a cada fase para determinar se o produto dessa fase satisfaz ao que foi requerido no início da fase visando responder a seguinte pergunta "estamos desenvolvendo o produto corretamente?". Entre suas atividades pode-se citar: inspeções (verificacação estática) e testes (verificação dinâmica).

Validação é o processo de se avaliar um software, durante ou após o desenvolvimento, para determinar se o produto satisfaz aos requisitos do cliente, visando responder a seguinte pergunta: "estamos desenvolvendo o produto correto?". Entre suas atividades pode-se citar: homologação, testes de aceitação (beta) e revisões.

Questão 14

Inspeção: estática
Execução simbólica: dinâmica
Revisão técnica: estática

Critérios de correção

Questão 1

Grafo: 8 pontos; cobertura de nós: 8 pontos; cobertura de ramos: 8 pontos.

Quem deu condições (p.ex. "par e maior que 100") em vez de valores concretos (p.ex. 102) perdeu 2 pontos.

Quem não entendeu que cada if correspodia a dois nós, por causa da condição composta, perdeu 2 pontos.

Questão 2

Quem fez a tabela corretamente ganhou 1,11 ponto por cada coluna correta. Mas quem fez uma linha só para a saída perdeu 2 pontos. Quem não colocou a saída de 0% perdeu 2 pontos. Quem não colocou a parte de baixo da tabela perdeu 3 pontos.

Quem fez a tabela transposta, ou seja, linhas em lugar de colunas e vice-versa, perdeu pelo menos 1 ponto. Se não explicitou a saída, perdeu 3 pontos.

Quem fez um arranjo 3x3 ou outra forma tabular que não a correta, ficou com 3 pontos nesta questão.

Questão 3

Grafo: 5 pontos; def: 5 pontos; c-uso: 5 pontos; p-uso: 5 pontos.

Quem não entendeu que while correspodia a dois nós, por causa da condição composta, perdeu 2 pontos.

Falta de ramo voltando do end-while ao while: perdeu 2 pontos; if seguidos em vez de aninhados: perdeu 2 pontos.

Questão 4

Quem não modelou o número de vezes que se pode errar perdeu 3 pontos. Quem modelou isto usando variáveis perdeu 1 ponto.

Quem não rotulou arestas perdeu 1 ponto.

Quem omitiu nós os arestas perdeu no máximo 1 ponto por elemento omitido.

Questão 5

Quem: 2 pontos; Métricas: 2 pontos. Foi considerado correto dizer que as métricas podem ser definidas no documento de Critérios de Teste.

Questão 6

O conceito de qualidade de software valia 2 pontos. Os fatores valiam 2 pontos.

Quem acertou 5 fatores ganhou 2; quem acertou de 1 a 4 fatores ganhou 1 ponto; quem não acertou enhum fator ganhou 0.

Para acertar um fator era necessário dar-lhe o nome correto e a definição correta. Exemplos de nomes incorretos que NÃO foram aceitos: acertividade (isto acho que nem existe no dicionário), legibilidade (o correto é manutenibilidade ou flexibilidade), durabilidade, adaptabilidade (o correto é portabilidade).

Questão 7

Cada item valia 1,5 pontos.

Questão 8

Acertou tudo: 2 pontos; acertou alguns: 1 ponto; errou tudo: 0 pontos.

Questão 9

Acertou tudo: 2 pontos; acertou alguns: 1 ponto; errou tudo: 0 pontos.

Questão 10

Acertou tudo: 2 pontos; acertou algo: 1 ponto; errou tudo: 0 pontos.

Questão 11

Quem observou que complexidade ciclomática está relacionado ao código e que testes caixa-preta não utilizam código, acertou. Quem não fez esta observação, errou.

Questão 12

Acertou tudo: 2 pontos; acertou algo: 1 ponto; errou tudo: 0 pontos.

Questão 13

Vaidação: 1 ponto; verificação: 1 ponto.

Questão 14

Acertou: 1 ponto; errou:0 pontos.

MC626 Home