Fundação CECIERJ - **Vice Presidência de Educação Superior a Distância**

Curso de Tecnologia em Sistemas de Computação
Disciplina: Engenharia de Software
AP2 2° semestre de 2013 - GABARITO

Nome –

Assinatura –

Observações:

1. Prova sem consulta e sem uso de máquina de calcular.
2. Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
3. Você pode usar lápis para responder as questões.
4. Ao final da prova devolva as folhas de questões e as de respostas.
5. Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

---

1) Na notação UML, uma classe é representada por um retângulo dividido em 3 regiões. Que informações colocamos em cada região da classe? Como estas informações se relacionam ao princípio do encapsulamento? (valor: 2,0 pontos; máximo: 10 linhas).

R: A divisão no topo da classe mantém o nome da classe; a divisão central mantém os atributos da classe; a última divisão mantém os métodos da classe. O princípio de encapsulamento determina que todas as classes devem ter uma interface bem definida, por onde ela oferece os seus serviços. Esta interface é composta pelos métodos públicos da classe, que são os únicos capazes de acessar os métodos privados e os atributos.

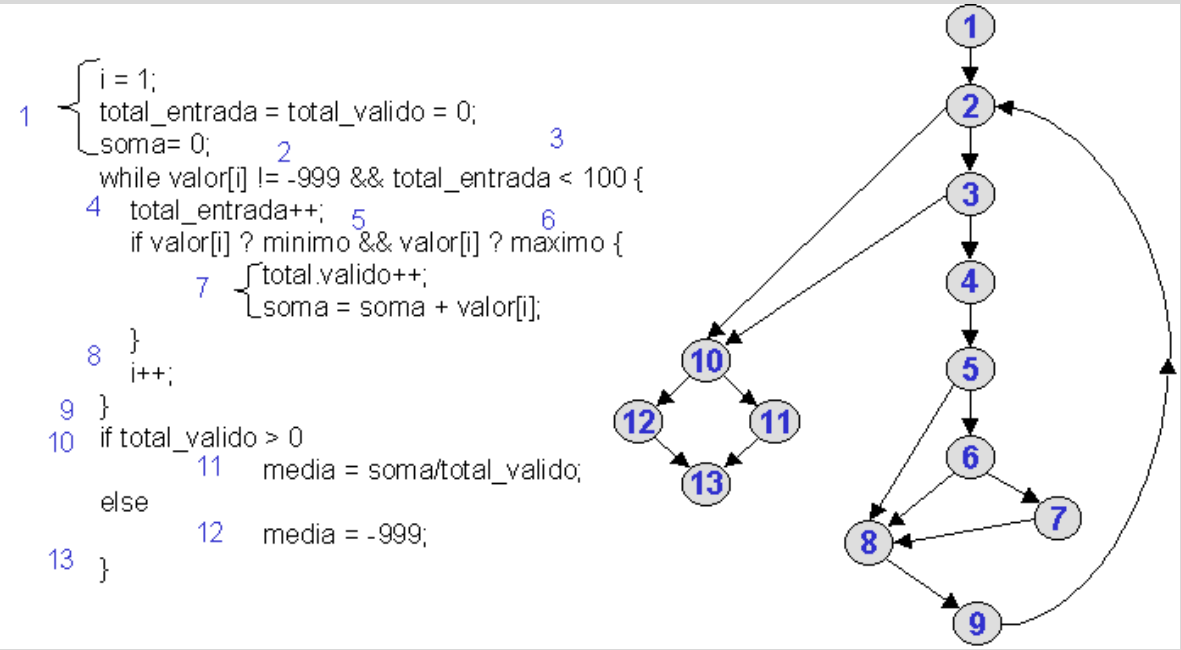
2) Explique a relação entre as atividades de planejamento, execução e controle na gerência de um projeto de desenvolvimento de software. Qual é o papel de cada uma destas fases e como elas se relacionam entre si? (valor: 2,0 pontos; máximo: 10 linhas).

R: O planejamento envolve a definição, revisão e manutenção de uma organização de trabalho para a realização do projeto. Ele precede a atividade de execução. A execução, por sua vez, consiste na coordenação das pessoas e recursos necessários para a execução do plano construído durante o planejamento. Por fim, o controle envolve a monitoração e medição do progresso do projeto, para garantir que seus objetivos serão atingidos. Ele acontece em paralelo com a execução.

3) O que pode indicar a métrica Complexidade Ciclomática (*McCabe*)? Que artefato de software (dê um exemplo) pode ser utilizado para observá-la e como ela pode ser obtida ou calculada? (valor: 2,0 pontos)

R: Esta métrica fornece uma medida quantitativa da complexidade lógica de um programa. No contexto do teste estrutural, seu valor define o número de caminhos independentes no grafo de programa e nos fornece o número máximo de casos de teste que garantem que todos os comandos tenham sido executados pelo menos uma vez.

O artefato que pode ser usado para visualiza-la é o grafo de programa. Um exemplo pode ser visto abaixo:



Fonte: www.dimap.ufrn.br/~jair/ES/c8.html

Seu calculo se dá através da identificação do número de regiões no grafo de fluxo do programa, através da formula

V(G)=E-N+2

E: número de arcos

N: número de nós

ou, V(G) = P +1

P: número de nós predicados (decisões)

4) Defina os casos de teste para estes requisitos usando a técnica de **particionamento por equivalência** (Valor 2 pontos):
Em uma indústria química o controle da temperatura e pressão de um determinado processo depende de dois fatores: a temperatura de ebulição do produto sendo processado e de um coeficiente Z. As

entradas do programa de controle são a temperatura de ebulição do produto em graus Celsius limitada ao intervalo [80;210] e o coeficiente Z limitado ao intervalo [0.2;2.5]. O sistema de controle deve suspender o processo se a temperatura ou a pressão atingirem valores críticos.
O cálculo do valor crítico da temperatura é obtido pela fórmula: tk = te*2.
O valor crítico da pressão pela fórmula:

i. pk = tk * Z, quando te for menor que 105;
ii. pk = te * Z nos demais casos.
iii. Onde, tk=temperatura crítica, te=temperatura de ebulição, pk=pressão crítica.

Como este software foi construído visando *testabilidade*, é possível entrar com um terceiro e quarto valores correspondentes à temperatura e a pressão em um dado instante. Nesse caso, o software efetua os cálculos e imprime uma mensagem sinalizando se interromperia ou não o processo.

R:

Classes de equivalência:

Entrada	válida	válida	inválida	inválida
te	80 <= te <105	105 <= te < =210	te < 80	te > 210
Z	0.2 <= Z <= 2.5		Z < 0.2	Z > 2.5
Comportamento	tk = te * 2 pk = tk * Z	pk = te * Z	--	--
Pk (calculado)	32 <= pk < 542	21 <= pk < 525	pk < 21	pk > 525

Casos de Teste: CT:{temperatura, pressão, comportamento}

CT1: {79, 15, Não interromper}

CT2: {215, 526, Interromper}

CT3: { 100, 55, Não Interromper }

CT4: {100, 21, Não Interromper }

CT5: {100, 545, Interromper}

CT6: {150, 18, Não Interromper}

CT7: {150, 160, Não Interromper}

CT8: {150, 530, Interromper}

5) Defina ***falta***, ***erro***, ***falha*** e ***defeito*** e indique quais técnicas podem ser usadas para identificar cada um deles. (10 linhas, valor 2 pontos).

R:

Falta: Deficiência mecânica ou algorítmica que se ativada pode levar a uma falha.

Erro: Item de informação ou estado de execução inconsistente.

Falha: Evento notável onde o sistema viola suas especificações.

Defeito: termo genérico para identificar faltas, erros e/ou falhas. Defeitos podem ser classificados como de omissão, inconsistência, fato incorreto, ambiguidade e informação estranha.

Técnicas que podem ser utilizadas para identificar defeitos (faltas, erros): Inspeção de software (técnicas Ad-hoc, checklist ou de Leitura).

Técnica que pode ser utilizada para revelar falhas e consequentemente indicar defeitos: Testes de software