UNIVERSIDADE AbERTA
www.uab.pt

Ministério da Ciência, Tecnologia e Ensino Superior

U.C. 21025

Desenvolvimento de Software

3 de Julho de 2013

INSTRUÇÕES

PARA A RESOLUÇÃO DO EXAME, ACONSELHA-SE QUE LEIA ATENTAMENTE O SEGUINTE:

1. Para a resolução deste **p – Fólio** aconselha-se que:
2. Verifique o exemplar que lhe foi entregue e, no caso de estar incompleto ou com qualquer deficiência, dirija-se ao professor vigilante.
3. O **p-fólio** é composto por 3 questões.
4. O teste termina com a palavra **FIM**.
5. Utilize, sempre, uma letra legível e não use uma caneta de outra cor que não seja o preto ou o azul - as respostas a lápis não serão consideradas.
6. Tenha em atenção que o **p-Fólio** tem a duração de 1 hora e 30 minutos.

Bom Trabalho!

1º Questão (8 Valores)

Nas questões seguintes indique na folha de teste a opção correta. Cada resposta correta vale 0,5 valores, cada resposta incorreta desconta 0,1 valores. Cada questão não respondida vale 0 valores.

Suponha que se está a desenvolver um sistema para uma biblioteca. Este possui 3 módulos, um está relacionado com as entradas e saídas de livros; outro está relacionado com o controlo de inventário; o último módulo produz relatórios. Durante o desenvolvimento do sistema ocorreram vários problemas:

1. Considere as seguintes afirmações:
   A – No código para calcular as taxas dos livros em atraso de devolução a variável ***fine_total*** não foi inicializada.
   B- Quando o operador tenta adicionar um novo livro ao inventário o sistema faz **shut down.**

   a) A é um defeito e B é um erro.
   b) é um erro e B uma falha.
   c) A é um defeito e B uma falha
   d) A é um erro e B um defeito
   e) A é uma falha e B um erro

2. Em qual das seguintes fases de desenvolvimento de software não participa o programador:

   a) Desenho do programa
   b) Teste de unidades
   c) Implementação
   d) Teste de integração
   e) Nenhuma das anteriores

3. Quais dos seguintes excertos podem ser considerados requisitos funcionais válidos:

   a) Quando o pagamento estiver completo o sistema, deve responder da seguinte forma: se o utilizador pagou diretamente ao funcionário, ou pagou na bomba mas não quer recibo, o sistema volta ao estado inicial, caso contrário imprime um recibo.
   b) Deve ser mantido um registo para cada funcionário da gasolineira. Este deve conter o seu nome, apelido, e número de funcionário. Os registos são guardados em listas ligadas.
   c) Após o cliente escolher um modo de pagamento na bomba, o sistema deve verificar se o input é valido (um número entre 1 e 3).
   d) Apenas a) e b) são verdadeiras
   e) Apenas a) e c) são verdadeiras

4. "Após o processo de pagamento da gasolina estar completo, a informação relevante deve ser adicionada um ficheiro de log".

   a) O requisito deve ser reescrito; pois está incorreto.
   b) O requisito deve ser reescrito; pois está ambíguo ou inconsistente.
   c) O requisito deve ser reescrito ; pois é irrealista.
   d) O requisito deve ser reescrito ; pois não é verificável.
   e) O requisito está correto.

5. “O sistema deve ser construido de forma a ser fácil implementar novas funcionalidades no futuro”.

a) O requisito deve ser reescrito; pois está incorreto.
b) O requisito deve ser reescrito; pois está ambíguo ou inconsistente.
c) O requisito deve ser reescrito ; pois é irrealista.
d) O requisito deve ser reescrito ; pois não é verificável.
e) O requisito está correto.

6. Relativamente à coesão de um componente, quando as partes têm de alterar o mesmo conjunto de dados estamos em presença de uma coesão:

a) Funcional
b) Comunicacional
c) Sequencial
d) Procedimental
e) Coincidental

7. Numa função de um componente de software, uma variável não foi inicializada de forma própria. Qual dos seguintes testes irá provavelmente expor este defeito:

a) testes unitários
b) testes de integração
c) testes de aceitação
d) testes de instalação
e) testes de performance

8. Numa gasolineira o sistema deve permitir ao utilizador decidir se quer o recibo imprimido ou não. No entanto a função de impressão não foi implementada. Qual dos seguintes testes irá provavelmente expor este defeito:

a) testes unitários do componente A
b) testes de integração dos componentes A e B
c) testes de aceitação
d) testes de instalação
e) testes de performance

9. Uma companhia de cartões de crédito fez um upgrade ao seu sistema para permitir maior segurança nos cartões de crédito. Esta alteração conduz a uma modificação ligeira do tipo de dados que têm de ser enviados. Esta situação:

a) deve conduzir a uma manutenção corretiva
b) deve conduzir a uma manutenção adaptativa
c) deve conduzir a uma manutenção perfectiva
d) deve conduzir a uma manutenção preventiva
e) não requer qualquer tipo de manutenção.

10. Foi adicionado um serviço adicional aos clientes de uma gasolineira. Estes podem agora alugar lugares de estacionamento. Esta situação:

a) deve conduzir a uma manutenção corretiva

b) deve conduzir a uma manutenção adaptativa
c) deve conduzir a uma manutenção aperfeiçoativa
d) deve conduzir a uma manutenção preventiva
e) não requer qualquer tipo de manutenção.

11. Suponha que se inseriram 100 defeitos num software, durante a fase de testes a equipa descobriu 60 desses defeitos e mais 60 defeitos diferentes. Qual o número de defeitos que ainda falta encontrar:

a) 7,5
b) 10
c) 40
d) 50
e) 60

12. Suponha que o código anterior fui dado a uma segunda equipa de testes. Esta encontrou 80 defeitos dos 100 colocados propositadamente. Encontrou ainda mais 70 defeitos. 90 dos defeitos encontrados por esta equipa foram também encontrados pela equipa anterior. Utilizando os números da segunda equipa. Qual o número de defeitos que ainda falta encontrar:

a) 0
b) 17,5
c) 20
d) 87,5
e) 60

13. Qual a eficiência do segundo grupo de teste:

a) 30%
b) 40%
c) 60%
d) 85%
e) É impossível determinar a partir da informação fornecida.

Pergunta anulada por troca das opções de resposta

14. Qual a eficiência do primeiro grupo de teste:

a) 25%
b) 42%
c) 64%
d) 75%
e) É impossível determinar a partir da informação fornecida.

Pergunta anulada por troca das opções de resposta

15. Baseados na eficiência dos dois grupos de teste, qual a estimativa para o número total de defeitos.

a) 200
b) 100
c) 78
d) 32

e) É impossível determinar a partir da informação fornecida.

16. Os ambientes de desenvolvimento modernos ao completarem parte do código automaticamente têm tendência a evitarem:

a) Falhas.
b) Enganos.
c) Erros.
d) Defeitos.
e) Crash.

2º Questão (3 Valores)

Atendendo às normas de escrita de código em anexo (igual às disponibilizadas no espaço da unidade curricular), escreva uma função que dados dois argumentos inteiros positivos, retorne **true** caso estes sejam números primos gémeos e **false** no caso contrário. Dois números inteiros são primos gémeos se a diferença entre eles for igual a 2 e forem ambos primos. Por exemplo, 11 e 13 são primos gémeos.

```
bool primosGemeos(int a, int b)
{
 for (int i=a/2; i>1; i--)
 if (a%i==0)
   return false;
      for (int j=b/2; j>1; j--)
      if (b%j==0)
   return false;
      if ((b-a)==2 || (b-a)==-2)
   return true;
 else
   return false;
}
```

3º Questão (1 Valor)

O que entende por rejuvenescimento de software. Que estratégias se podem utilizar?

Ver sebenta página 100

Anexo – Norma de Escrita de Código

1. Formatação
1.1 Indentação: Cada linha de código deve começar por 4 espaços por cada bloco de código em que estiver inserido. É aceitável em vez de 4 espaços utilizar outro valor, desde que seja sempre o mesmo para todo o ficheiro.
1.2 Tamanho das linhas: As linhas tanto código como de comentários não devem exceder os 80 caracteres. Caso tal não seja possível, a linha de código deve ocupar mais que uma linha. Conta para efeitos de indentação a segunda linha e seguintes, como se estivessem dentro de mais um bloco de código. Uma linha de comentário que continue na linha seguinte mantém a indentação.
1.3 Argumentos: As listas de argumentos, tanto na definição como na utilização, não são considerados blocos de código. Assim devem ser escritos na mesma linha, se possível, caso contrário aplica-se o ponto 1.2.

1.4 Início e fim de bloco: A marca de início e de fim de bloco de código deve utilizar uma linha para o início e outra para o fim do bloco. As linguagens funcionais podem ignorar esta regra, uma vez que em geral o primeiro elemento do bloco é um identificador e convém que fique na mesma linha que a abertura de bloco.

2. Comentários

2.1 Localização: Os comentários devem anteceder o código que comentam. Pode-se colocar pequenos comentários ao bloco na própria linha de início ou fim de bloco de código.

2.2 Comentário de classe: No início de cada classe deve existir um comentário de classe. Neste comentário deve conter informação sumária sobre a classe. Dá-se liberdade de incluir mais ou menos campos conforme apropriado, no entanto o comentário de classe deve ter obrigatoriamente os campos (inglês/português):

- Author/Autor - uma ou mais pessoas que contribuíram para a classe
- Last change/Última modificação - data da última modificação
- Description/Descrição - descrição da estrutura de dados e breve descrição dos algoritmos envolvidos, esclarecendo o que os algoritmos fazem e não como fazem

2.3 Comentário de método: Cada método deve ter um comentário inicial, idêntico ao comentário de classe, acrescentando:

- Descrição do algoritmo, não só o que faz mas também como o faz;
- A complexidade temporal do algoritmo, sempre que possível. No início dos blocos de código pode-se colocar também a complexidade temporal do bloco de código;
- Pré e pós-condições de utilização do método, por exemplo, os elementos de uma lista estarem ordenados;
- Incluir eventuais restrições de utilização, por exemplo, o valor máximo para cada argumento do método para o qual o método foi testado e funciona bem. Após serem realizados testes, os resultados dos testes devem ser também incluídos.

2.4 Comentário de linha: Devem ser colocados comentários de linha apenas em zonas complexas, de forma a complementar a descrição do algoritmo feita no comentário de método. Os comentários de linha devem assumir que o leitor conhece a linguagem de programação, pelo que não devem repetir o que está no código.

3. Nomes

3.1 Variáveis iteradoras inteiras: Os nomes das variáveis devem começar por letras minúsculas. As variáveis iteradoras inteiras devem ser chamadas i, j e k. Devem ser reutilizadas, devendo o k ser utilizado apenas se o i e j estiverem em utilização. Caso sejam necessárias mais que três variáveis iteradoras, adicionar um número à letra i (i1, i2, ...). Caso as variáveis iteradoras sejam reais ou de outra natureza, utilizar as letras l, m e n.

3.2 Variáveis não iteradoras: Para as variáveis não iteradoras deve-se utilizar nomes elucidativos. Na junção de dois nomes, a letra inicial do segundo nome deve ser maiúscula. A primeira letra deve ser sempre minúscula. Caso não existam nomes elucidativos, utilizar as letras u, v e w para os valores inteiros, x, y e z para os valores reais, a, b e c para nomes (strings), e p, q e r para apontadores ou variáveis de outra natureza.

3.3 Constantes e macros: Constantes ou macros devem ser escritos em maiúsculas, e as palavras separadas por um "underscore": '_'.

3.4 Estruturas e classes: Os nomes das estruturas ou classes têm de ter sempre nomes elucidativos e devem começar por T se é um tipo abstracto de dados, S se for uma estrutura, e C se for uma classe no geral, D se for uma classe para caixas de diálogo de uma interface gráfica. Na junção de dois nomes, a letra inicial de cada nome deve ser maiúscula.

3.5 Métodos: Os nomes dos métodos devem ter sempre nomes elucidativos. No caso de programação não orientada por objectos, o nome das funções e procedimentos deve ser antecedida do nome ou iniciais do módulo a que pertencem. Na junção de dois nomes, a letra inicial de cada nome deve ser maiúscula. Não utilizar o mesmo nome para funções ou métodos com argumentos diferentes, a não ser que as funções ou métodos sejam apenas diferentes versões. Em princípio, o nome de um procedimento deve conter um verbo no infinitivo, enquanto que o nome de uma função deverá ter o nome da grandeza que calcula, não necessitando de verbo. As funções com resultado booleano devem conter um verbo no presente ou adjectivo de forma a serem utilizadas dentro de condicionais.

4. Estilo

4.1 Blocos de código: Não abrir um bloco de código para um só comando, excepto para evitar confusão por exemplo numa cadeia if/else.

4.2 Repetições: Não repetir uma sequência de comandos. Muitas vezes por facilidade ao escrever código, repetem-se comandos quando poderiam ser postos dentro de um ciclo.

4.3 Tamanho dos métodos: Não fazer métodos demasiado grandes (mais de 100 linhas de código). Nesse caso agrupar comandos relacionados em métodos, mesmo que o método venha a ser chamado uma só vez.
4.4 Comandos numa linha: Iniciar sempre uma nova linha para um comando, ou seja, não colocar antes de um comando um condicional, ciclo, ou outro comando.

FIM