Universidade Estadual Vale do Acaraú
Curso de Ciências da Computação
Disciplina de Laboratório de Programação
Pf. Paulo Regis Menezes Sousa

Exercćicios

Questão 1. 1 P.

Crie uma estrutura representando um aluno de uma disciplina. Essa estrutura deve guardar as seguintes informações: o número de matrícula do aluno (um inteiro de 4 dígitos), sexo (masculino ou feminino), seu índice de rendimento (um inteiro de 0 a 100) e se é ou não a primeira vez que o aluno faz a disciplina.

Escreva um programa que mostre o tamanho em bytes dessa estrutura e realize os ajustes necessários para que ela não possua mais que 4 bytes.

Questão 2. 4 P.

Crie um programa de gerenciamento de estoque. O programa deve exibir um menu com todas as opções do usuário.

1P Cadastrar um produto

1P Excluir um produto (selecionado pelo código)

1P Listar os produtos

+1P Um ponto a mais se a listagem for em ordem alfabética

Um produto tem os seguintes atributos:

- Código (é gerado pelo programa)
- Nome
- Marca
- Preço

Crie uma estrutura para representar os produtos e um vetor com capacidade para 100 produtos. Use o vetor como base de dados para o programa.

Questão 3. 2 P.

1P Crie uma estrutura representando uma hora. Essa estrutura deve conter os campos hora, minuto e segundo. Agora, escreva um programa que leia um vetor de cinco posições dessa estrutura.

1P Depois de lido o vetor busque e imprima a maior hora.

Exemplo de Entrada	Exemplo de Saída
15 30 0 9 59 1 5 1 30 23 59 59 12 40 12	Maior hora: 23 59 59

Questão 4. 2 P.

O jogo UNO possui cartas numéricas e cartas de ação, as cartas de ação podem ser de dois tipos: cartas de ação com cores e as cartas de ação curinga que tem todas as cores ao mesmo tempo.

1P Crie um tipo `Carta` adequado para representar uma carta de UNO e em seguida declare e inicialize um vetor com 6 posições desta estrutura.

1P Crie uma função `int avaliarJogada(Carta a, Carta b)`, esta função deve comparar se a carta `a` e a carta `b` são compatíveis. As cartas são compatíveis quando:

- São cartas da mesma cor
- Pelo menos uma delas é um curinga

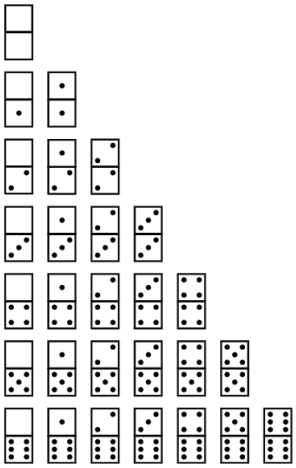
Se as cartas são compatíveis a função deve retornar 1, do contrário deve retornar 0. Use as cartas do vetor para testar a função.

Questão 5. 3 P.

1P Crie um tipo `Peca` para representar uma peça de dominó, a estrutura deve poder armazenar dois números inteiros que representam as extremidades (chamaremos de ponta A e ponta B) de uma peça.

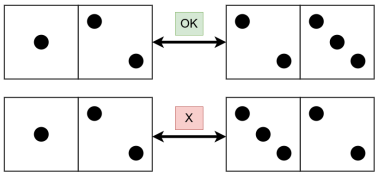
1P Faça o tipo `Peca` usar apenas 1 byte de memória para armazenar os dois valores das pontas da peça.

1P Inicialize um vetor do tipo `Peca` para armazenar as 28 peças do dominó e inicialize-as com os valores corretos das peças e imprima no console as peças criando o mesmo padrão mostrado na figura abaixo.

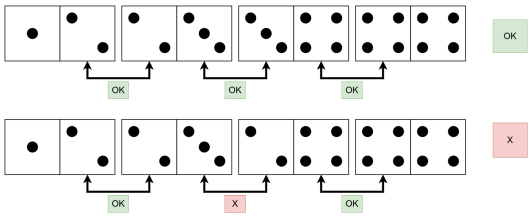


Questão 6. 3 P.

1P Crie uma função `int avaliarJogada(Peca p1, Peca p2)` que receba duas peças como parâmetro. A função deve comparar se a peça `p2` pode ser colocada ao lado da ponta B da peça `p1` (veja a figura a baixo) e retornar 1 caso seja uma jogada válida ou 0 do contrário.



2P Crie uma função **`int avaliaSequencia(Peca p[], int tam)`**, onde `p` é um vetor de peças e `tam` o seu tamanho. Use as funções anteriores para verificar se a sequência de peças é válida.



A função retorna 1 se todas as ligações forem válidas e 0 caso contrário.

Questão 7. 3 P.

2P Crie uma função `map` com a seguinte assinatura:

```
float *map(float v[], float z[], int l, float (*fun)(float));
```

onde, `l` é o tamanho dos vetores `v` e `z` e `fun` um ponteiro para função.

A função `map` deve iniciar o vetor `z` com o resultado da função `fun` aplicada à respectiva posição de `v`: `z[i] = fun(v[i])`.

1P Peça ao usuário que preencha `v` e demonstre o funcionamento da função `map` usando funções diferentes como parâmetro.