Prova 2

Algoritmos e Estruturas de Dados I

Professor: Pedro O.S. Vaz de Melo

10 de junho de 2016

Nome: _____

escrevendo o meu nome eu juro que seguirei o código de honra

Código de Honra para este exame:

- Não darei ajuda a outros colegas durante os exames, nem lhes pedirei ajuda;
- não copiarei nem deixarei que um colega copie de mim;
- não usarei no exame elementos de consulta não autorizados.

Importante: nas questões a seguir, você vai implementar um simulador de compras de uma loja semelhante ao que foi passado em sala de aula. A diferença é que os produtos terão estoque limitado e clientes poderão realizar mais de uma compra. O seu programa deve sortear o número de clientes, número de produtos disponíveis na loja, estoque de cada produto, preço de cada produto, número de compras de cada cliente e produtos que cada cliente comprou. Depois disso, deve imprimir o faturamento da loja naquele dia. Para resolver as questões considere as definições dadas na última questão desta prova.

1. (*4 points*) Defina os dois novos tipos de dados descritos abaixo:

a. (*2 pts*) Defina um **novo tipo de dados** para representar um `Produto` e que deve ser capaz de armazenar as seguintes informações: id (um número inteiro), estoque (número de itens em estoque) e preço (com parte fracionária, ex: $9,99$).

b. (*2 pts*) Defina um **novo tipo de dados** para representar um `Cliente` e que deve ser capaz de armazenar as seguintes informações: `id` (um número inteiro), `ncompras` (número de produtos comprados pelo cliente), `carrinho` (lista de produtos comprados pelo cliente). Considere que a lista de produtos é um vetor que conterá os `id`s dos produtos comprados pelo cliente. Use a definição `MAXCOMPRAS` como limite deste vetor.

2. (*2 points*) Implemente uma função de nome `initCliente` que recebe um `Cliente` **por referência** e um identificador `cod` por valor. A função deve inicializar os campos `id` e `ncompras` do cliente que recebeu como parâmetro com os valores `cod` e 0, respectivamente.

3. (*4 points*) Implemente uma função de nome `initProduto` que recebe um `Produto` **por referência** e um identificador `cod` por valor. A função deve inicializar os campos do produto da seguinte forma: `id` deve receber `cod`, `estoque` deve receber um número aleatório entre 0 e `MAXESTOQUE` e `preco` um valor aleatório entre 1.00 e 100.00, de forma que os centavos também sejam preenchidos aleatoriamente.

4. (*4 points*) Implemente uma função de nome `estoqueLoja` que recebe como parâmetros um vetor de `Produto`s e a quantidade de produtos que a loja vende. A função deve retornar o número total de produtos que estão em estoque na loja. Protótipo:

```
int estoqueLoja(Produto p[], int n);
```

5. (*6 points*) Implemente uma função de nome `simulaCompras` que recebe como parâmetros um `Cliente` (por referência), um vetor de produtos, e o número de produtos diferentes que a loja vende. A sua função deve simular as compras do cliente. Primeiro, sorteie o número de produtos que o cliente comprará (0 a `MAXCOMPRAS`). Depois, sorteie códigos de produtos e adicione esses produtos no `carrinho` de compras do cliente, atualizando o campo `ncompras` do mesmo. Importante: você deve garantir que o cliente não compre um produto que não tenha em estoque. Além disso, você deve garantir que a sua função termine quando o número total de produtos em estoque na loja seja 0 (dica: use a função `estoqueLoja`).

6. (*5 points*)
Implemente uma função de nome `totalVendas` que calcula o faturamento total da loja ao fim da simulação. A função recebe um vetor de `Clientes` com as informações de compra de cada cliente, o número total de clientes que fez compras na loja, e o vetor de produtos contendo as informações dos produtos que a loja vende. Note que o vetor de produtos é indexado pelo identificador do produto, ex: `p[8]` contém as informações do produto de código 8.

```
float totalVendas(Cliente c[], int nc, Produto p[]);
```

7. (*3 points*)
Complete o código abaixo.

```
#include <stdio.h>
#include "prova2.h"
#define MAXCLIENTES 50 //numero maximo de clientes
#define MAXPRODUTOS 1000 //numero maximo de produtos diferentes que a loja vende
#define MAXESTOQUE 3 //estoque maximo de cada produto
#define MAXCOMPRAS 20 //numero maximo de produtos que cada cliente pode comprar

void main() {

   Cliente clientes[MAX_CLIENTES];
   Produto produtos[MAX_PRODUTOS];
   int nclientes, nprods, i;
   nclientes = rand()%MAX_CLIENTES;
   nprods = rand()%MAX_PRODUTOS;
   printf("\nnumero clientes: %d", nclientes);

   //inicializa cada cliente do vetor clientes
   for(i=0; i<nclientes; i++) {

      initCliente(____________________, i);
    }

   //inicializa cada produto do vetor produtos
   for(i=0; i<nprods; i++)

      initProduto(____________________, i);
   }

   //simula a compra de cada cliente do vetor clientes
   for(i=0; i<nclientes; i++) {

      simulaCompras(____________________, produtos, nprods);
   }
   float saldo = totalVendas(clientes, nclientes, produtos);
   printf("\nTotal vendas: %f", saldo);
}
```