INSTITUTO FEDERAL DA PARAÍBA.

CURSO DE BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO.

DISCIPLINA: ESTRUTURA DE DADOS E LABORATÓRIO DE ESTRUTURA DE DADOS.

ATIVIDADE AVALIATIVA 02 (parte da nota 3)

Questão 1: Responder a seguinte questão por meio de um programa em C.

Em uma pacata cidade do interior, o senhor Severino decidiu montar a própria biblioteca, já que coleciona vários livros desde sua juventude. Como ele não sabe programar, pediu ajuda ao neto para criar um programa que cadastre e ordene seus livros pelo código. Porém, seu neto ainda está no ensino fundamental, e como sabe muito pouco de programação, acabou criando um programa que somente cadastra os livros, mas não os ordena. Desse modo, o senhor Severino recorreu a você, pois sabe de suas habilidades com programação. Sua tarefa é simples: ordenar os cadastros dos códigos dos livros.

Entrada

A entrada contém vários casos de teste. Cada teste começa com um valor N ($1 \leq N \leq 1000$). Em seguida, N linhas terão os códigos dos livros, que estão sempre no formato "xxxx", isto é, não haverá o cadastro '1', por exemplo, mas "0001". Note que se trata de uma entrada em formato String.

Saída

Seu programa deverá imprimir o cadastro dos códigos ordenado.

Para ordenar você deve usar o método de ordenação por **seleção**. Nesse caso, adapte a implementação vista em sala para que possam ser ordenados Strings ao invés de números.

Veja os exemplos de entrada e saída a seguir:

Exemplo de Entrada	**Exemplo de Saída**
3 1233 0015 0100	0015 0100 1233
7 0752 1110 0001 6322 8000 6321 0000	0000 0001 0752 1110 6321 6322 8000

Questão 2: Considere um tipo que represente um funcionário de uma empresa, definido pela estrutura a seguir:

```
typedef struct funcionario {
  char nome[50];
  float valor_hora;
  int horas_mes;
} Funcionario;
```

Agora, faça o que se pede:

a) Usando a ordenação por **bolha**, implemente uma função que ordene um vetor de funcionários em ordem alfabética de nomes, seguindo o protótipo:

```
void ordem_alfabetica_func(int tam, Funcionario *vf);
```

b) Escreve uma função que faça uma **busca binária** em vetor de Funcionario, cujos elementos estão em ordem alfabética dos nomes dos funcionários. Essa função deve receber como parâmetro o número de funcionários, o vetor e o nome do funcionário que se deseja pesquisar, e deve ter como valor de retorno um ponteiro para a estrutura do funcionário procurado. Se não houver um funcionário com o nome procurado, a função deve retornar NULL. A função deve obedecer a seguinte assinatura:

```
Funcionario *busca_func(int n, Funcionario *vf, char *nome);
```

c) Escreva uma função que faz a mesma coisa que o item b) (busca binária pelo nome do funcionário), porém a busca deve ser realizada em um vetor de ponteiros para o tipo Funcionario, conforme a assinatura a seguir:

```
Funcionario *buscabin_func(int n, Funcionario *vf[], char *nome);
```

d) Escreva uma função que exiba os nomes dos três funcionários com maiores salários, sendo o salário de um funcionário igual ao número de horas trabalhadas no mês multiplicado pelo valor da sua hora de trabalho. Assuma que não existem dois funcionários com o mesmo salário. A função deve receber como parâmetros o número total de funcionários e o vetor que armazena ponteiros para a estrutura do tipo Funcionario, de acordo com a seguinte definição:

```
Funcionario *exibe_top3_salarios(int n, Funcionario *vf[], char *nome);
```

Obs: sua função deve primeiro ordenar o vetor em ordem decrescente de salários. Para isso, use a técnica de ordenação por inserção;

Escreva uma função main para testar todas as funções.

Continua na próxima página ...

Questão 3: Considere a definição de árvore binária de pesquisa usada em sala aula e a implementação das operações fornecidas pelo professor no moodle presencial:

```
typedef struct noarv {
  struct noarv *esq;
  struct noarv *dir;
  int dado;
} NoArv;

typedef struct arvore {
  NoArv *raiz;
} Arvore;

// + as operacoes de criar, exibir, inserir, remover e pesquisar
```

Agora, adapte a árvore de tal forma que o dado armazenado na árvore seja a estrutura Funcionario usada na questão anterior, acrescentando um campo para matricula para servir de chave. Adapte a implementação de todas as operações e crie um main para testar a funcionalidade da árvore.