Universidade Federal de Goiás
Introdução à Programação- 2023-1
Prova P5

Prof. Thierson Couto Rosa

Instruções

- Não é permitido utilizar celular, fone de ouvido, calculadora, pendrives e outros dispositivos eletrônicos distintos do computador.
- Os únicos sites que podem ser acessados durante a prova são: a) o site do Sharife, o site cplusplus e o site GDB online.
- Cópias de parte do código de colegas ou da internet implicarão em nota zero na questão onde a cópia ocorreu.

Sumário

1 Premiação Ao Mais Antigo - 4,0 pontos

Estamos chegando ao fim do ano, na época de Natal, e uma empresa resolveu prover uma série de premiações aos seus funcionários. Uma delas é a premiação aos três funcionários mais antigos da empresa. A empresa contratou você para escrever um programa que leia os dados dos funcionários da empresa e que os ordene em uma ordem tal que o funcionário mais antigo apareça em primeiro lugar, o segundo funcionário mais antigo apareça em segundo lugar.

Os dados de cada funcionário fornecidos a você são:

- o nome do funcionário (no máximo 50 caracteres seguidos por uma quebra de linha);
- ano da contratação (um números seguido por uma quebra de linha);
- uma linha com no máximo 200 caracteres contendo um breve histórico das ocupações desempenhadas pelo funcionário na empresa;
- o valor do salário atual do funcionário (um valor double seguido por uma quebra de linha);

Escreva um programa que leia os dados de vários funcionários da empresa, ordene os funcionários e imprima os dados dos três funcionários mais antigos na empresa.

Entrada

A primeira linha da entrada contém o número N, $0 > N \leq 200$, de funcionários da empresa. Em seguida, para cada um dos N funcionários, há quatro linhas, cada uma contendo um dado de um funcionário, conforme descrito acima.

Saída

A saída contém os dados **dos três funcionários** mais antigos, mostrados em ordem decrescente de antiguidade (o mais antigo primeiro). Para cada usuário, deve haver as seguintes linhas na saída:

Nome: *n*
Ano de contratacao: *a*
Ocupacoes: *o*
Salario: *s*
onde *n* é o nome do funcionário, *a*, é o ano em que o funcionário foi contratado, *o* é uma string contendo uma lista de ocupações desempenhadas pelo funcionário na empresa e *s* é o salário atual do funcionário. Separando as linhas referentes a um funcionário daquelas referentes ao próximo funcionário deve haver uma linha em branco.

Sugestão

Primeiramente escreva código somente para ler os dados e para mostra-los na tela para você se certificar que o programa está lendo corretamente antes de escrever código para a ordenação. A leitura da entrada é um pouco complicada pois há duas strings a serem lidas. Lembre-se de consumir os caracteres de quebra de linha que ficam no buffer do teclado antes de ler uma string.

Exemplo

Entrada
5 Antonio Vasconcelos Rogrigues 1978 Auxliar de escritorio, secretario, gerente administrativo 10000.00 Bruno Oliveira 2015 Servicos gerais 1000.00 Antonieta Pires 1968 Secretaria, gerente administrativa 20000.00 Maria Luiza de Souza 1975 Secretaria 12000.00 Leonardo Taveira 1980 gerente de RH 20000.00
Saída
Nome: Antonieta Pires Ano de contratacao: 1968 Ocupacoes: Secretaria, gerente administrativa Salario: 20000.00 Nome: Maria Luiza de Souza Data de contratacao: 1975 Ocupacoes: Secretaria Salario: 12000.00 Nome: Antonio Vasconcelos Rogrigues Data de contratacao: 1978 Ocupacoes: Auxliar de escritorio, secretario, gerente administrativo Salario: 10000.00

2 Premiação Ao Mais Antigo (versão 2) - 6,0 pontos

A empresa que te contratou para resolver o problema da questão 2 não está satisfeita em ter que informar o número de funcionários que ela possui, pois esse número varia de um ano para outro e é necessário contar os funcionários antes de entregar a lista ao seu programa. Ela quer que você escreva uma nova versão do programa que leia diretamente os dados dos funcionários e que pare quando encontrar fim de arquivo. O programa deve continuar gerando a mesma saída. A entrada é que mudou. Agora não tem mais um número inteiro como primeiro dado entrada. Veja o exemplo para ver como a entrada mudou. Repare que agora você não sabe mais quantos funcionários há na entrada. Gere uma nova versão do seu programa que atenda a empresa. Antes de terminar, o programa deve liberar toda área de memória, caso ele tenha alocado alguma. .

Os dados de cada funcionário fornecidos a você são:

- o nome do funcionário (no máximo 50 caracteres seguidos por uma quebra de linha);
- ano da contratação (um números seguido por uma quebra de linha);
- uma linha com no máximo 200 caracteres contendo um breve histórico das ocupações desempenhadas pelo funcionário na empresa;
- o valor do salário atual do funcionário (um valor double seguido por uma quebra de linha);

Escreva um programa que leia os dados de vários funcionários da empresa, ordene os funcionários e imprima os dados dos três funcionários mais antigos na empresa.

Entrada

A entrada é formada por zero ou mais quadruplas de linhas. Cada quádrupla contém uma linha para:

- o nome de um funcionário;
- o ano de contratação do funcionário;
- o histórico do funcionário;
- o salário do funcionário.

Saída

A saída contém os dados **dos três funcionários** mais antigos, mostrados em ordem decrescente de antiguidade (o mais antigo primeiro). Para cada usuário, deve haver as seguintes linhas na saída:

Nome: *n*
Ano de contratacao: *a*
Ocupacoes: *o*
Salario: *s*
onde *n* é o nome do funcionário, *a*, é o ano em que o funcionário foi contratado, *o* é uma string contendo uma lista de ocupações desempenhadas pelo funcionário na empresa e *s* é o salário atual do funcionário. Separando as linhas referentes a um funcionário daquelas referentes ao próximo funcionário deve haver uma linha em branco.

Exemplo

Entrada
Antonio Vasconcelos Rogrigues 1978 Auxliar de escritorio, secretario, gerente administrativo 10000.00 Bruno Oliveira 2015 Servicos gerais 1000.00 Antonieta Pires 1968 Secretaria, gerente administrativa 20000.00 Maria Luiza de Souza 1975 Secretaria 12000.00 Leonardo Taveira 1980 gerente de RH 20000.00
Saída
Nome: Antonieta Pires Ano de contratacao: 1968 Ocupacoes: Secretaria, gerente administrativa Salario: 20000.00 Nome: Maria Luiza de Souza Data de contratacao: 1975 Ocupacoes: Secretaria Salario: 12000.00 Nome: Antonio Vasconcelos Rogrigues Data de contratacao: 1978 Ocupacoes: Auxliar de escritorio, secretario, gerente administrativo Salario: 10000.00

3 Strcpy Irritante - Questão opcional (2,0 pontos)

A função `strcpy()` tem o seguinte protótipo:

```
char* strcpy ( char* destino, const char * origem );
```

A função tenta copiar a string cujo endereço está em `origem` para o vetor de caracteres cujo endereço está em `destino`. A função pode gerar um erro de execução (por invasão de memória) se o número de bytes do vetor cujo endereço está em `destino` for inferior ao tamanho da string que está em `origem`, incluindo o caractere de fim de cadeia. A função retorna o mesmo endereço que recebe em `destino`.

Juca Nerdinho é muito bom aluno da disciplina Introdução à Programação do INF-UFG, porém, quando a disciplina chegou no tópico Strings, Juca começou a se sair mal nas listas de exercícios desse tópico. Ele fica irritado pois, toda vez que precisa copiar uma string para a outra, ele tem que garantir que a primeira string possui espaço suficiente para comportar a segunda string. Ele se esquece de fazer essa checagem e seus programas frequentemente causam o maldito erro de *segmentation fault* (invasão de memória). Juca Nerdinho não queria mais usar a *irritante* `strcpy` da stdlib.h e resolveu criar sua própria função de cópia de strings a qual ele resolveu denominar `strcpynerd` em homenagem a si mesmo. Juca resolveu que a função deveria ter o seguinte protótipo:

```
char* strcpynerd(char** destino, char* origem);
```

Repare que o primeiro parâmetro, *destino* tem um tipo de dados diferente do primeiro parâmetro *destino* da `strcpy`. Na `strcpynerd`, o primeiro parâmetro é um ponteiro para ponteiro de `char`. Isso se faz necessário porque Juca quer que, em alguns casos, a sua função mude o endereço da string destino. Um exemplo de quando essa situação é necessária ocorre quando a string apontada por *origem* é maior do que a primeira string. Nesse caso é necessário realocar espaço na memória suficiente para armazenar uma cópia da string apontada por origem e pode ser que o endereço da string destino necessite ser mudado por essa razão. Juca quer que a nova função tenha o seguinte funcionamento:

1. se o endereço armazenado em `*destino` for NULL (zero), a função deve alocar espaço suficiente para caber a string apontada por `origem` (incluindo o caracter \0) e armazenar o endereço do espaço alocado em `*destino`;

2. se a string cujo endereço está em `*destino` tem tamanho inferior ao da string apontada por `origem`, a função deve realocar espaço e o endereço para o espaço alocado de ser armazenado em `*destino`;

3. se o tamanho da string cujo endereço está em `*destino` for maior ou igual ao tamanho da string apontada por `origem`, a função deve apenas executar a cópia e não deve alocar ou liberar memória;

4. a função não pode em hipótese alguma chamar a função `strcpy`, pois Juca, que é faixa preta em caratê, ficaria ainda mais irritado se soubesse que a função proposta por ele usaria a `strcpy`;

5. ao tentar alocar ou realocar espaço e não houver memória suficiente para essas operações, a função não deve terminar o programa, mas sim, atribuir NULL a `*destino` e retornar NULL. Essa ação é importante pois transfere para o código que chamou a função a responsabilidade de tomar a decisão em caso de falta de memória.

O problema de Juca é que o professor dele ainda não explicou alocação dinâmica de memória. Ele precisa que você implemente a função `strcpynerd` para ele.

Para testar a função, ele quer que você além de implementar a função, escreva um programa que declare uma variável ponteiro para char, inicialmente com NULL, por exemplo `char *dest=NULL`, e uma string de 5000 caracteres para armazenar linhas lidas na entrada (ex. `char linha[5000]`). Seu programa deve ler linhas na entrada enquanto houver strings para serem lidas. Para cada string lida o programa deve chamar a função `strcpynerd` para copiar a string lida em uma área de memória. O endereço para a string copiada

deve ficar armazenado na variável ponteiro para char (na variável `dest` do exemplo) se a operação de cópia for executada com sucesso. Após a cópia, seu programa deve imprimir a string copiada, cujo endereço está na variável ponteiro para char (a variável `dest`, no exemplo). Se a função `strcypnerd` retornar NULL, seu programa deve emitir a frase "Nao ha memoria suficiente para copiar a string. Encerrando o programa." e deve terminar o programa imediatamente.

Entrada

A entrada contém várias linhas com texto.

Saída

A saída é formada pelas mesmas linhas da entrada

Exemplo

Entrada

`Ola, eu sou o Juca Nerdinho,`
`por favor, implemente a funcao strcpynerd para mim.`
`Nao aguento mais receber o erro segmantation fault nos meus programas`
`que usam a funcao strcpy da stdlib.h.`
`Que funcao irritante!!!!!`

Saída

`Eu sou o Juca Nerdinho,`
`por favor, implemente a funcao strcpynerd para mim.`
`Nao aguento mais receber o erro segmantation fault nos meus programas`
`que usam a funcao strcpy da stdlib.h.`
`Que funcao irritante!!!!!`

Observação

Essa questão é opcional. Se o aluno resolver essa questão corretamente pode conseguir até 2 pontos que podem ser adicionados a essa mesma prova ou a uma prova anterior em que o aluno não consegui nota máxima.

```
void* realloc (void *ptr, size_t size);
```

A função muda o tamanho do bloco de memória apontado pelo ponteiro `ptr`. A função pode move o bloco de memória para uma nova área de memória (cujo endereço de memória é retornado pela função). Nesse caso, os dados armazenados no bloco apontado por `ptr` são copiados para a nova área de memória.