Ifes Campus Serra
BSI – Bacharelado de Sistemas de Informação
Estrutura de Dados, 2024-1

Avaliação 2: TADLista

Regras para Prova

- Todos os códigos produzidos como resposta às questões da prova deverão conter **apenas** o subconjunto de comandos da linguagem **C ANSI utilizada em sala de aula**. Caso contrário, haverá descontos na pontuação, e em alguns casos implicará em **nota zero** para a prova.
- **Todos os nomes fornecidos pelos enunciados devem ser mantidos**: arquivos de código, arquivos de dados, funções e tipos de dados. Caso contrário, a nota da prova será zero.
- Todo o código entregue como resposta ao(s) enunciado(s) deve ser compilável pelas versões do **gcc** estudadas e utilizadas em aula.
- Comandos e **funções C não vistas** em aula NÃO serão aceitas durante a correção;
- **A prova é individual;**
- Todos oss **códigos idênticos** entre alunos receberão nota da prova igual a **zero**.
- O material entregue para a tarefa deverá estar em um arquivo compactado chamado **ed-p2.zip**.
- Não será corrigido o material de prova entregue em arquivos que não sejam **zip** ou cujo o nome não seja aquele especificado no item anterior.
- O arquivo **paraprova2.zip** contém todo o material de prova necessário para a confecção das questões.
- A **questão única** deverá ser resolvida em um arquivo nomeado **resposta.c.**
- **USE O tadlista.h do professor e o tadlista.c da sua autoria.**

Correção da Prova

- Arquivo resposta.c **não compila** (erros de compilação): nota zero.
- Binário gerado **falha na execução** e produz um erro incortonável: nota zero.
- Binário é gerado mas a sua execução **não resolve** o problema do enunciado: nota zero.
- **tadlista.h foi alterado**: nota zero.
- Na **aplicação**, **apenas** os tipos **Lista e t_dado podem ser usados**, caso contrário, nota zero.
- Manipulação de string não usa string.h, nota receberá desconto proporcional à quantidade de locais do programa onde a manipulação de string não ocorreu com string.h.
- Binário foi gerado, executa e produz uma saída coerente para os dados de entrada fornecidos no enunciado: nota será definida pela lógica, estrutura do programa e manuseio correto dos conceitos de ponteiro e alocação dinâmica.
- **Lógica e estrutura do programa tem prioridade sobre a qualidade do resultado.**

Em caso de dúvidas, contacte o Professor via o fórum ou chat do ava.

Questão Única – Enunciado (30 Pontos)

A interpretação do enunciado e figuras faz parte da prova.

RESOLVA A PROVA USANDO O TADLISTA.H FORNECIDO PELO PROESSOR E O TADLISTA.C FORNECIDO POR VOCÊ.

No enunciado a seguir, as seguintes notações serão usadas:

[1,2,3,4]: lista do tipo Lista (tadlista desenvolvido em aula) (notação Python).

<nome, idade, altura>: struct contendo os campos nome, idade e altura (notação ed.bsi).

<nome, .., altura>: struct , notação resumida (notação ed.bsi).

Utilizando o tadlista (talista.h, tadlista.o), construa um arquivo chamado r**esposta.c** contendo todas as funções e tipos de dados pedidos no(s) enunciado(s):

Um arquivo chamado bdceps.txt possui os dados de 25 endereços fictícios. Os dados em cada linha (separados por vírgula) são: cep, numero da casa, nome da família. Neste arquivo existe mais de uma linha com o mesmo cep, e elas estão espalhadas pelo arquivo. Ceps possuem quantidades de endereços diferentes.

Sabendo disto, escreva um programa C (**resposta.c**) contendo o que é pedido nos itens a seguir:

a) Declare na área de **typedef** um tipo endereco 1 (t**_endereco1**) capaz que armazenar os dados de um endereço (1 linha de bdceps.txt);

b) Declare na área de **typedef** um tipo endereco 2 (**t_endereco2**) que possua a seguinte estrutura: <cep, Lista>: cada variável do tipo t_endereco2 contem 1 cep e uma lista de todos os endereços (do tipo t_endereco1) que possuem o mesmo cep.

Exemplo: <cep, [<t_endereco1>, <t_endereco_1>, .. , <t_endereco1>]>

c) Construa a função ***Lista loadbdceps(char *nomearq)***: a função lê **UM** arquivo do tipo bdceps.txt (linha a linha) e retorna uma Lista de estruturas do tipo t_endereco1. O arquivo deve ser aberto e fechado nesta função. O nome do arquivo é definido somente em ***main()***.

d) Construa a função ***Lista converte(Lista lst1)***: função recebe uma lista do tipo daquela do item **(c)** e retorna uma lista do tipo [<t_endereco2>, <t_endereco2>, .., <t_endereco2>]: uma lista de enderecos do tipo 2 (t_endereco2).

Exemplo ilustrativo:

[<29000111, [<29000111, 201, Silvas>, <29000111, 345, Monteiros>] **>**,

<39000110, [<39000110, 151, Caldas>, <39000110, 45, Garcias>, <39000110, 345, Leites>] **>]**

e) Construa a função ***void print_enderecos_2(Lista lst)*****:** a função recebe uma lista de endereços do tipo do item **(d)** e a exibe na tela: o cep deve ocupar sozinho uma linha; abaixo da linha do cep, os respectivos endereços (dados do tipo t_endereco1) cada um ocupando a sua respectiva linha. Uma linha em branco deve separar os diferentes ceps.

f) Construa a função ***int main()***: a função deve invocar as funções dos itens anteriores em uma sequencia tal que processa os dados do arquivo **bdceps.txt** e exibe o seu conteúdo na tela com o a organização definida por meio da função ***print_enderecos_2(..).***

Sugestão

Trate a lista do tipo 2 como um dicionário python: construa uma função auxiliar que receba a lista do tipo 2 e um valor de cep, e retorne a lista de endereços deste cep (ver estrutura t_endereco2).

Material fornecido para a prova

- Enunciado ed-prova-2.pdf .
- Arquivos de dados bdceps.txt.
- Arquivo de cabeçalho taddlista.h (use o seu tadlista,c)

Material de Apoio

- Còdigos no repositório Replit.com: @ernanifo

Entrega

- Compacte os arquivos resposta.c, bdceps.txt, tadlista.h e tadlista.c (sua autoria) em um único arquivo .zip chamado <sua matricula bsi>.zip, envie o arquivo para a tarefa no ava.
- **Formato rar não será aceito.**

Boa Prova!!