Instituto de computação

TCC 00347 Nome: _____
Prof. Aline Paes Avaliação - Programação Estruturada

- Atente para modularizar seu código com funções, usar nomes de variáveis e funções que tenham significado e tornar seu código o mais legível possível.
- Caso não saiba resolver todos os subproblemas envolvidos na solução, deixe ao menos indicada a abstração para a(s) função(ões) que resolveria(m) o(s) subproblema(s). Com isso, seu programa pode receber uma pontuação parcial.
- Faz parte da sua avaliação entender o enunciado. Durante a avaliação, nenhuma dúvida será respondida.
- A duração da avaliação é de três horas.
- Não é preciso dizer que em qualquer tentativa de cola ou plágio, a nota será zero
- Você deve inserir seu arquivo de implementação .c no classroom. A não inserção da sua resposta significa que eu não terei com o que te avaliar.

Considere para as questões abaixo a existência de um arquivo de interface *palavras.h* e outro *palavras.c*, ainda incompleto, conforme exibidos após as questões.

1. **(2,0 pontos)** Implementar a função *ler_com_duplicado*, que receberá uma string indicando o nome de um arquivo texto e fará com que cada palavra do arquivo texto seja um elemento da lista encadeada. Assuma que a separação das palavras no texto seja apenas por espaços em branco entre elas.

2. **(2,5 pontos)** Implementar a função *ler_sem_duplicado*, que receberá uma string indicando o nome de um arquivo texto e fará com que cada palavra do arquivo texto seja um elemento único na lista encadeada. Ou seja, se uma palavra apareceu antes na lista encadeada, ela não deve aparecer de novo. Para cada palavra, a lista encadeada deve conter a quantidade de vezes que ela apareceu no texto, usando o campo *qtd*. Assuma que a separação das palavras no texto seja apenas por espaços em branco entre elas.

3. **(2,5 pontos)** Ordenar a lista encadeada por quantidade de vezes que as palavras aparecem na lista. O algoritmo de ordenação deve ser implementado por você e deve ser ou o *quicksort* ou o *mergesort*. Para facilitar, é permitido que você use vetores para auxiliar a ordenação.

4. **(2,0 pontos)** Implementar uma função *remover_palavra*, considerando duas situações: (i.) se a quantidade de vezes que a palavra apareceu no texto é maior que 1, a função deve apenas decrementar a quantidade. Caso a quantidade seja 1, a palavra deve ser removida de fato da lista encadeada.

5. **(1,0 pontos)** Implementar a função liberar, que exclui todos os elementos da lista encadeada e a própria lista.

```c
typedef struct palavra Palavra;

typedef struct palavras ListaPalavras;

ListaPalavras* ordenar (ListaPalavras* ps);

ListaPalavras* criar ();

void liberar (ListaPalavras* ps);

ListaPalavras* ler_com_duplicado(ListaPalavras* ps, char* nome_arquivo);

ListaPalavras* ler_sem_duplicado(ListaPalavras* ps, char* nome_arquivo);

ListaPalavras* remover(ListaPalavras* ps, char* palavra);
```

Listing 1: palavras.h

```c

#include <stdio.h>
#include <stdlib.h>
#include "palavras.h"

typedef struct palavra {
    char palavra[30];
    int qtd;
}Palavra;

typedef struct el_palavra {
    Palavra* pl;
    struct el_palavra* prox;
}ElementoPalavra;

typedef struct palavras {
    ElementoPalavra* inicio;
}ListaPalavras;

ListaPalavras* criar(){
    ListaPalavras* lp = (ListaPalavras*) malloc(sizeof(ListaPalavras));
    lp->inicio = NULL;
    return lp;
}
```

Listing 2: palavras.c