

Avaliação Final
Curso: Engenharia da Computação
Disciplina: Estruturas de Dados
Prof. Jarbas Joaci de Mesquita Sá Junior
Universidade Federal do Ceará – UFC/Sobral

Entrega: 16/02/2022 via e-mail para

Obs.:

- O trabalho é **individual** e **não** será recebido após a data mencionada;
- Preferencialmente fazer o trabalho usando a IDE Dev-C++;
- Enviar **todos** os arquivos do projeto, exceto os executáveis (.exe). Organizar os arquivos nas pastas q1, q2.
- O uso da diretiva `#include` sem um header file (.h) implicará nota zero no código. Por exemplo, **não** usar `#include "nomearquivo.c"`

1ª) Implemente o Tipo Abstrato de Dados (TAD) “lista.h” (ver slides sobre Listas Encadeadas) e acrescente as seguintes funções:

a) função para criar uma lista que **some** uma lista L_2 com uma lista L_1 (as listas L_1 e L_2 não devem ser alteradas) de tal forma que a lista resultante tenha a seguinte sequência: 1º elemento de L_1 + 1º elemento de L_2 , 2º elemento de L_1 + 2º elemento de L_2 , e assim por diante. Se as listas tiverem tamanhos diferentes, a parte excedente da lista maior (ou seja, para a qual não há pareamento) deverá ser concatenada na nova lista. Essa função deve obedecer ao protótipo:

```
Lista* lst_soma_p(Lista* l1, Lista* l2);
```

Por exemplo, se lista $L_1 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 9 \rightarrow //$ e lista $L_2 \rightarrow 2 \rightarrow 7 \rightarrow //$, a chamada `Lista* L3 = lst_soma_p(L1, L2)` gera a lista $L_3 \rightarrow 5 \rightarrow 12 \rightarrow 6 \rightarrow 9 \rightarrow //$.

b) função para criar uma lista que é a concatenação de duas listas L_1 e L_2 sem os valores primos (as listas L_1 e L_2 não devem ser alteradas). Essa função deve obedecer ao protótipo:

```
Lista* lst_conc_sem_primos(Lista* l1, Lista* l2);
```

Por exemplo, se lista $L_1 \rightarrow 3 \rightarrow 7 \rightarrow 2 \rightarrow 4 \rightarrow //$ e lista $L_2 \rightarrow 7 \rightarrow 9 \rightarrow //$, a chamada `Lista* L3 = lst_conc_sem_primos(L1, L2)` gera a lista $L_3 \rightarrow 4 \rightarrow 9 //$.

A seguir, execute o seguinte programa.

```
#include <stdio.h>
#include <stdlib.h>
#include "lista.h"
```

```
int main(void) {
```

```
    Lista* l1 = lst_cria();
```

```

l1 = lst_inserir_ordenado(l1,21);
l1 = lst_inserir_ordenado(l1,42);
l1 = lst_inserir_ordenado(l1,33);
l1 = lst_inserir_ordenado(l1,65);
l1 = lst_inserir_ordenado(l1,11);
l1 = lst_inserir_ordenado(l1,17);

Lista* l2 = lst_cria();
l2 = lst_inserir_ordenado(l2,23);
l2 = lst_inserir_ordenado(l2,40);
l2 = lst_inserir_ordenado(l2,7);

Lista* l3=lst_soma_p(l1,l2);
lst_imprime(l3);

Lista* l4=lst_conc_sem_primos(l1,l2);
lst_imprime(l4);

lst_libera(l1); lst_libera(l2);
lst_libera(l3); lst_libera(l4);

system("PAUSE");
return 0;
}

```

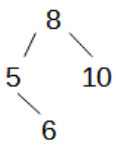
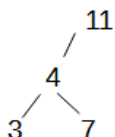
2. Implemente a TAD “arvb.h” (Árvore Binária de Busca) e acrescente as seguintes funções:

a) função que crie uma lista formada pelos elementos em comum de duas árvores (as árvores não devem ser alteradas). Os elementos da lista devem estar em ordem crescente. Essa função deve obedecer ao protótipo:

```
Lista* arvs_elem_comuns_lista(ArvB* a, ArvB* b);
```

b) função que crie uma lista encadeada formada pelos elementos de duas árvores concatenados por camada (as árvores não devem ser alteradas). Essa função deve obedecer ao protótipo:

```
Lista* arvs_elem_camadas_lista(ArvB* a, ArvB* b);
```

Por exemplo, se arv_1 é , e arv_2 é 

então a chamada `Lista* L = lst_elem_camadas_lista(arv_1, arv_2)` gera a lista $L_3 \rightarrow 8 \rightarrow 11 \rightarrow 5 \rightarrow 10 \rightarrow 4 \rightarrow 6 \rightarrow 3 \rightarrow 7 \rightarrow //$.

A seguir, execute o seguinte programa.

```
#include <stdio.h>
#include <stdlib.h>
#include "arvb.h"
#include "lista.h"

int main(void){

    ArvB* arv_1 = arvb_cria_vazia();
    arv_1=arvb_inserere(arv_1,43);
    arv_1=arvb_inserere(arv_1,51);
    arv_1=arvb_inserere(arv_1,21);
    arv_1=arvb_inserere(arv_1,4);
    arv_1=arvb_inserere(arv_1,45);
    arv_1=arvb_inserere(arv_1,29);
    arv_1=arvb_inserere(arv_1,3);
    arv_1=arvb_inserere(arv_1,23);

    ArvB* arv_2 = arvb_cria_vazia();
    arv_2=arvb_inserere(arv_2,45);
    arv_2=arvb_inserere(arv_2,26);
    arv_2=arvb_inserere(arv_2,23);
    arv_2=arvb_inserere(arv_2,31);
    arv_2=arvb_inserere(arv_2,47);

    Lista* L1=arvs_elem_comuns_lista(arv_1,arv_2);
    lst_imprime(L1);

    Lista* L2=arvs_elem_camadas_lista(arv_1,arv_2);
    lst_imprime(L2);

    lst_libera(L1); lst_libera(L2);
    arvb_libera(arv_1); arvb_libera(arv_2);

    system("'PAUSE'");
    return 0;
}
```