Fatec São Caetano do Sul Antônio Russo

ESTRUTURA DE DADOS

ADSMA3

São Caetano do sul

2021

Fatec São Caetano do Sul Antônio Russo

NOME: João Pedro De Araujo silva RA:1680482011017

NOME: Lucas Henrique Andrade da Rocha RA: 1680482011001

Exercício de Estrutura de Dados como nota parcial da N2 do 3º semestre orientado pela Professora Rosana Maria Traversa

São Caetano do sul

2021

EXERCÍCIOS PARA A P2

Os exercícios (funções) solicitados (VALEM 0.5 PONTO CADA) e devem ser realizados em duplas (identifique a matrícula, nome e turma de cada integrante da dupla) na data estipulada para a entrega, em arquivo único no formato PDF.
As funções devem ter o código fonte comentado.
Os exercícios que não estiverem dentro dos requisitos acima serão desconsiderados.

1. Uma lista duplamente encadeada guarda informações de nome e data de nascimento de diversas pessoas. Nesta lista os nomes são únicos, mas ela não foi criada de forma ordenada por nomes. Escreva uma função **recursiva** que faça a procura sequencial de um nome nessa **lista.** São passados como parâmetros para esta função um ponteiro para o **elemento final** da lista, e um nome (n). A função deve informar a data de nascimento correspondente ao **nome encontrado**, ou informar **não encontrado** se toda lista for percorrida sem sucesso.

Considere: typedef struct lista{
```
    char nome[21];
    char  nasc[11];
    struct lista *ant;
    struct lista *prox;
}Lista;
```
Protótipo: **`void busca(Lista *last, char *n)`**

```
////////////////////////////////////////////////////////
void busca(Lista *last, char *n) {
     if(last!=NULL){
          if(!strcmp(last->nome,n))// comparacao entre
string
               printf("%s\t",last->nasc); // printa a data
          else
               busca(last->prox,n); // vai pro proximo
     }
     else
          printf("nome nao encontrado");// se não encontrar
nada printa a mensagem
}
////////////////////////////////////////////////////////
```

2. Escreva uma função que receba um vetor de números reais, já preenchido e seu tamanho. Grave todo o vetor, em uma única operação, em um arquivo tipo binário de nome "Reais.bin".

Protótipo: **`void gravaNumeros (double *num, int tam)`**

Considere:

```
typedef struct arv{
   int matricula; // a árvore está organizada pela matricula
   float nota;
   struct arv *left, *right;
   }Arv;
```

```
////////////////////////////////////////////////////////////////
void gravaNumeros (double *num, int tam){
    int i;
     FILE *fp;
     fp=fopen("Reais.bin","w+b"); // abre o arquivo
     fwrite(num, sizeof(int),tam,fp); /// grava o arquivo
     for(i=0;i<=tam;i++){
          printf("\n %.lf", num[i]); // printa oq ta no arquivo
      }
     fclose(fp);
}
////////////////////////////////////////////////////////////////
```

3. Escreva uma **função** que mostre a matrícula e a nota dos elementos da árvore que tenham nota igual a nota passada por parâmetro.

//Protótipo: **`void mostra(Arv *T, float nota)`**

```
////////////////////////////////////////////////////////////////
void mostra(Arv *T, float nota){
     if(T!=NULL){
          if(T->nota==nota){ // comparação entre a nota dada e a
nota que esta na arvore
               printf("%d\t",T->matricula); // printa
          }
          mostra(T->esq,nota); // percusos pra esquerda
          mostra(T->dir,nota); // percusos pra direita
     }
}
////////////////////////////////////////////////////////////////
```

4. Escreva uma função que retorne o número de elementos desta árvore que tenham nota maior que a nota passada por parâmetro. (Não use variáveis globais)

//Protótipo: **`int conta(Arv *T, float nota)`**

```
////////////////////////////////////////////////////////////////
int conta(Arv *T, float nota){
     int cont=0;
     if(T!=NULL){
          if(T->nota > nota) // compara de a nota dada é menor
que a nota da árvore
               cont= cont+1;
          cont= cont+conta(T->esq,nota); // vai pra esquerda e
verifica
          cont= cont+conta(T->dir,nota); // vai pra direita e
verifica
     }
     return cont; // retorna cont
}
////////////////////////////////////////////////////////////////
```