1) Implemente uma Árvore para armazenar números inteiros com as seguintes funções:

```
NoArvore* arvore_criaVazia(void);
NoArvore* arvore_cria(int info, NoArvore *sae, NoArvore *sad);
int arvore_vazia(NoArvore *a);
void arvore_libera(NoArvore *a);
void arvore_imprime(NoArvore *a);
```

Além das funções acima, a Árvore também deve implementar e exportar as seguintes funções:

nos_folhas – A função recebe uma árvore binária e retorna o número de folhas dessa árvore (o total de nós que não tem nenhum filho). Essa função tem o seguinte protótipo:

```
int nos_folhas(NoArvore *a);
```

nos_intermediarios – A função recebe uma árvore binária e retorna a quantidade de nós que não são folhas (o total de nós que tem pelo menos um filho). Essa função tem o seguinte protótipo:

```
int nos_intermediarios(NoArvore* a);
```

arvore_alturaMaior – A função recebe uma árvore binária e um valor inteiro. A função deve verificar se a altura da árvore é maior que o valor inteiro recebido por parâmetro. A função deve retornar 1 se a altura é maior, e 0 caso contrário. A função tem o seguinte protótipo:

```
int arvore_alturaMaior(NoArvore *a, int altura);
```

Após implementar as funções, utilize a seguinte função principal para testar o seu programa:

```c
#include <stdio.h>
int main()
{
    int n;
    NoArvore *arvore = arvore_Cria(1, arvore_Cria(2,
arvore_Cria(4,arvore_criaVazia(),arvore_criaVazia()),arvore_Cri
a(5,arvore_criaVazia(),arvore_criaVazia())),arvore_Cria(3,arvor
e_criaVazia(),arvore_Cria(6,arvore_criaVazia(),arvore_criaVazia
())));
    printf("Arvore:\n\n");
    arvore_imprime(arvore);
    printf("\n");

    n = nos_folhas(arvore);
    printf("Total Folhas: %d\n", n);

    n = nos_intermediarios(arvore);
    printf("Total Intermediarios: %d\n", n);

    n = arvore_alturaMaior(arvore, 2);
    printf("Altura da arvore eh maior que 2? %s\n", n == 1?
"Sim":"Nao");

    return 0;
}
```

2) Implemente um TAD (Tipo Abstrato de Dados) Árvore Binária de Busca (ABB) para armazenar ponteiros para estruturas do tipo Aluno:

```
struct aluno {
    int matricula
    char *nome;
    float media;
};
```

Os nós da árvore binária de busca são definidos pela seguinte estrutura:

```
struct noABB {
    Aluno *info;
    struct noABB *esq;
    struct noABB *dir;
};
```

O TAD ABB deve implementar as seguintes funções:

abb_insere – A função deve criar e inserir um novo nó na ABB respeitando a ordenação por média dos nós. Essa função tem o seguinte protótipo:

```
NoABB *abb_insere(NoABB *a, int mat, char *nome, float media);
```

abb_imprime – A função de imprimir na tela todas as informações (matricula, nome e média) de todos os alunos existentes na ABB em ordem crescente de média. Essa função tem o seguinte protótipo:

```
void abb_imprime(NoABB *a);
```

abb_libera – a função deve liberar da memória todos os nós da ABB. Essa função tem o seguinte protótipo:

```
void abb_libera(NoABB *a);
```

Além das funções normais, o TAD ABB também deve implementar e exportar as seguintes funções:

abb_alunoComMaiorMedia – A função recebe uma árvore binária de busca e imprime as informações do aluno com a maior média. A sua função deve levar em consideração a ordenação da árvore binária de busca (isto é, não percorra a árvore inteira, siga apenas o caminho que leva ao nó com a maior média). Essa função tem o seguinte protótipo:

```
void abb_alunoComMaiorMedia(NoABB *a);
```

abb_contaAprovados – A função recebe uma árvore binária de busca e retorna o número de alunos aprovados (isto é, alunos com média maior ou igual a 5.0). A sua função deve levar em consideração a ordenação da árvore binária de busca (isto é, não percorra a árvore inteira se não for necessário). Essa função tem o seguinte protótipo:

```
int abb_contaAprovados(NoABB *a);
```

abb_imprimeAprovados – A função recebe uma árvore binária de busca e imprime a matricula, nome e a média dos alunos aprovados em ordem DECRESCENTE. A

sua função deve levar em consideração a ordenação da árvore binária de busca (isto é, não percorra a árvore inteira se não for necessário). Essa função tem o seguinte protótipo:

```
void abb_imprimeAprovados(NoABB *a);
```

Após implementar as funções, utilize a seguinte função principal para testar o seu programa:

```c
#include <stdio.h>
int main(void) {
    NoABB *abb = abb_cria_vazia();
    abb = abb_insere(abb, 456124, "Pedro Duarte", 7.5);
    abb = abb_insere(abb, 453984, "Ana Silva", 8.8);
    abb = abb_insere(abb, 365597, "Joao Filho", 2.5);
    abb = abb_insere(abb, 687451, "Maria Gomes", 10.0);
    abb = abb_insere(abb, 364512, "Silvio Lins", 4.8);
    abb = abb_insere(abb, 984544, "Marcia Morais", 7.8);
    abb = abb_insere(abb, 698421, "Bruno Rodrigues", 2.0);
    abb = abb_insere(abb, 784512, "Thais Silva", 6.5);
    abb = abb_insere(abb, 694231, "Ricardo Costa", 9.5);
    abb = abb_insere(abb, 126411, "Julia Neves", 8.0);

    printf("Alunos em ordem crescente de media:\n");
    abb_imprime(abb);

    printf("\nAluno com maior media:\n");
    abb_alunoComMaiorMedia(abb);

    printf("\nNumero de alunos aprovados: %d\n", abb_contaAprovados(abb));

    printf("\nAlunos aprovados em ordem decrescente: \n");
    abb_imprimeAprovados(abb);

    return 0;
}
```