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 e q2.
- O uso da diretiva `#include` sem um header file (`.h`) implicará nota zero no código. Por exemplo, **não** usar `#include "nomearquivo.c"`

1. Implemente a TAD "arvb.h" (Árvore Binária de Buscas) e acrescente as seguintes funções:

a) função que retorne a quantidade de folhas de uma árvore binária de busca que possuem o campo `info` com `n` divisores positivos. Essa função deve obedecer ao protótipo:

```
int folhas_ndivp(ArvB* a, int n);
```

b) função que retorne a quantidade de nós de uma árvore binária de busca que possuem os dois filhos (campos `dir` e `esq` diferentes de `NULL`). Essa função deve obedecer ao protótipo:

```
int dois_filhos(ArvB* a);
```

c) função que, dada uma árvore binária de busca, retorne a quantidade de nós cujas subárvores esquerda e direita tenham igual altura. Essa função deve obedecer ao protótipo:

```
int nos_igual_altura(ArvB* a);
```

d) função que compare se duas árvores binárias de busca são iguais. Essa função deve obedecer ao protótipo:

```
int arv_iguais(ArvB* a, ArvB* b);
```

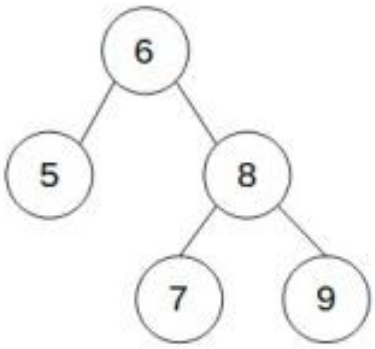
Obs: 1 – verdadeiro; 0 – falso.

e) função que imprima os elementos de uma árvore binária de busca por níveis. Essa função deve obedecer ao protótipo:

```
void impressao_arv_niveis(ArvB* a);
```

Por exemplo, na árvore da figura abaixo, a impressão deve ser:

6 – nível 0
5, 8 – nível 1
7, 9 – nível 2



A seguir, execute o seguinte programa.

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


int main(void){

  Arv* arvA = arvb_cria_vazia();
  arvA=arvb_insere(arvA,33);
  arvA=arvb_insere(arvA,5);
  arvA=arvb_insere(arvA,21);
  arvA=arvb_insere(arvA,4);
  arvA=arvb_insere(arvA,45);
  arvA=arvb_insere(arvA,28);
  arvA=arvb_insere(arvA,3);
  arvA=arvb_insere(arvA,2);
  arvA=arvb_remove(arvA,4);
  printf(''Qtd de nós dois filhos %d\n'',dois_filhos(arvA));
  printf(''Nós igual altura %d\n'',nos_igual_altura(arvA));
  printf(''Folhas: info com 6 div. %d\n'',folhas_ndivp(arvA,6));

  Arv* arvB = arvb_cria_vazia();

  arvB=arvb_insere(arvB,8);
  arvB=arvb_insere(arvB,9);
  arvB=arvb_insere(arvB,11);

  Arv* arvC = arvb_cria_vazia();

  arvC=arvb_insere(arvC,9);
  arvC=arvb_insere(arvC,8);
  arvC=arvb_insere(arvC,11);

  Arv* arvD = arvb_cria_vazia();

  arvD=arvb_insere(arvD,8);
  arvD=arvb_insere(arvD,9);
  arvD=arvb_insere(arvD,11);

  printf(''------\n'');
```

```
  impressao_arv_niveis(arvA);
  printf(''------\n'');

  int comp = arv_iguais(arvA,arvB);
  printf(''arvores iguais %d\n'',comp);
  printf(''------\n'');
  int comp = arv_iguais(arvB,arvC);
  printf(''arvores iguais %d\n'',comp);
  printf(''------\n'');
  int comp = arv_iguais(arvB,arvD);
  printf(''arvores iguais %d\n'',comp);

  arvb_libera(arvA);
  arvb_libera(arvB);
  arvb_libera(arvC);
  arvb_libera(arvD);

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

2. Implemente os algoritmos **BubbleSort**, **InsertionSort**, **QuickSort, MergeSort** e **HeapSort** e calcule o tempo de cada um para ordenar vetores de tamanho 10^2, 10^3, 10^4, 10^5 e 10^6 com os elementos dispostos de três formas: crescente, decrescente, e aleatória. Elabore um relatório com os dados obtidos.

Obs.: O tempo deve ser dado em milissegundos.