MINISTÉRIO DA EDUCAÇÃO
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ
CURSO : Analise e desenvolvimento de Sistemas
DISCIPLINA : Estrutura de Dados 2

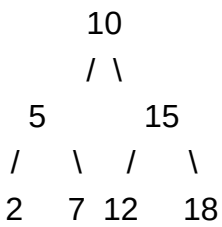
Nome:_____

Prova 1 ED2 – 23.05.2025

GABARITO:
MARQUE AQUI SUAS RESPOSTAS:

QUESTÕES	**RESPOSTAS**				
1.1 ***(0.5 pt)***	**A**	**B**	**C**	**D**	**E**
1.2 ***(0.5 pt)***	**A**	**B**	**C**	**D**	**E**
1.3 ***(0.5 pt)***	**A**	**B**	**C**	**D**	**E**
1.4 ***(0.5 pt)***	**A**	**B**	**C**	**D**	**E**
2 ***(1 pt)***	**A**	**B**	**C**	**D**	**E**
3 ***(1 pt)***	**A**	**B**	**C**	**D**	**E**
4.1 ***(1 pt)***	**A**	**B**	**C**	**D**	**E**
4.2 ***(1 pt)***	**A**	**B**	**C**	**D**	**E**
5 ***(1 pt)***	**A**	**B**	**C**	**D**	**E**
6 ***(1 pt)***	**A**	**B**	**C**	**D**	**E**

1. Sobre a árvore apresentada, selecione a alternativa correta para os percursos abaixo: (2.0 pts)



1.1. Percurso em Profundidade IN-ORDEM	A.(x) 2, 5, 7, 10, 12, 15, 18 B.( ) 5, 2, 7, 10, 12, 15, 18 C.( ) 10, 2, 5, 7, 12, 15, 18 D.( ) 2, 7, 5, 12, 10, 15, 18

1.2. Percurso em Extensão ou Largura	A.(x) 10, 5, 15, 2, 7, 12, 18 B.( ) 10, 15, 5, 2, 7, 12, 18 C.( ) 10, 5, 2, 7, 15, 12, 18 D.( ) 10, 5, 15, 7, 2, 12, 18

1.3 Percurso em Profundidade POS-ORDEM	A.(x) 2, 7, 5, 12, 18, 15, 10 B.( ) 2, 5, 7, 12, 18, 15, 10 C.( ) 2, 7, 5, 10, 12, 18, 15 D.( ) 7, 2, 5, 12, 18, 15, 10

1.4. Percurso em Profundidade PRE-ORDEM	A.(x) 10, 5, 2, 7, 15, 12, 18 B.( ) 10, 5, 15, 2, 7, 12, 18 C.( ) 10, 5, 2, 7, 15, 18, 12 D.( ) 10, 5, 7, 2, 15, 18, 12

2. Sobre o código abaixo: (1.0 pt)

```
stack<No*> pilha;
No *p = raiz;
string v;
if (p!=0){
    pilha.push(p);
    while (!pilha.empty()) {
        p=pilha.top();
        cout<<pilha.top()->nome<<endl;
        pilha.pop();
        if (p->right !=0)
            pilha.push(p->right);
        if (p->left != 0)
            pilha.push(p->left);
    }
}
```

a.( ) Está sendo realizado o percurso em extensão. O algoritmo é não-recursivo e usa uma pilha.

a. (x) Está sendo realizado o percurso em profundidade PRÉ-ORDER. O algoritmo énão-recursivo e usa uma pilha.

c.( ) Está sendo realizado o percurso em profundidade POS-ORDER. O algoritmo é não-recursivo e usa uma pilha.

d.( ) Está sendo realizado o percurso em profundidade PRÉ-ORDER. O algoritmo é recursivo e usa uma pilha.

3. São VERDADEIRAS as seguintes afirmativas: (1.0 pt)

<A> O processo de busca em um árvore é muito mais rápido do que o processo de busca em listas ligadas. No entanto, quando a árvore encontra-se ASSIMETRICA a eficiência do processo de busca pode ser comprometida. V

<B> Para calcular a quantidade total de nós em um árvore cheia, é possível elaborar o cálculo da quantidade total de nós a partir da altura da árvore. v

<C> Embora cada procedimento recursivo possa ser convertido em uma versão iterativa, a conversão nem sempre é uma tarefa trivial. Em alguns casos pode envolver explicitamente a manipulação de uma pilha. v

<D> Nem todo procedimento recursivo pode ser convertido em uma versão iterativa. f

<E> Todo procedimento recursivo pode ser transformado em uma versão iterativa, embora, em alguns casos, essa conversão possa ser complexa e exigir mais código. A recursão geralmente usa a pilha de chamadas para armazenar estados intermediários, enquanto a versão iterativa precisa de estruturas como pilhas, filas ou variáveis auxiliares para replicar esse comportamento. V
<F> A recursão é usualmente menos eficiente do que seu equivalente iterativo, mas, as vezes, se ela apresenta uma vantagem em clareza, legibilidade e simplicidade do código, a diferença no tempo de execução entre as duas versões pode ser desprezada. v

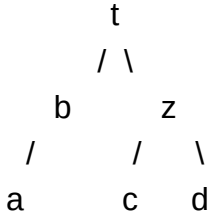
a. (x) <A>, <B>,<C>,<E>,<F>

b. ( ) Todas são verdadeiras

c. ( ) <A>, <B>,<C>,<F>

d. ( ) NDA

4. Considere, respectivamente, a árvore e o trecho de código abaixo: (2.0 pts)




```
typedef struct arv {
        char info;
        struct arv
        *esq; struct
        arv *dir;
}Arv;
Arv *arvore(char x,Arv *e,Arv *d){
        Arv*novo=(Arv*)malloc(sizeof(Arv));
        novo->esq=e;
        novo->dir=d;  novo->info=x;   return novo;
}
 main(){


}
```

4.1. Para preeencher a função “main()” de maneira que o programa construa a árvore apresentada, marque a alternativa correta:

a. ( )

```
main(){
    Arv *t = arvore('t',NULL,NULL);
    Arv *m = arvore('m',0,0);
    Arv *c = arvore('c',0,0);
    Arv *d = arvore('d',0,m);
   Arv *z = arvore('z',c,d);
   Arv *b = arvore('b',t,0);
  Arv *a = arvore('a',b,z);
}
```

c.( )

```
main(){
    Arv *a = arvore('a',NULL,NULL);
    Arv *m = arvore('m',0,0);
    Arv *c = arvore('c',0,0);
    Arv *d = arvore('d',0,m);
   Arv *z = arvore('z',c,0);
   Arv *b = arvore('b',a,0);
  Arv *t = arvore('t',b,z);
}
```

b.(x)

```
main(){
    Arv *a = arvore('a',NULL,NULL);
    Arv *m = arvore('m',0,0);
    Arv *c = arvore('c',0,0);
    Arv *d = arvore('d',0,m);
   Arv *z = arvore('z',c,d);
   Arv *b = arvore('b',a,0);
  Arv *t = arvore('t',b,z);
}
```

d.( ) NDA

4.2. Associe:

(1) percurso em profundidade IN-ORDER (2) percurso em extensão (3) percurso em profundidade POS-ORDER (4) percurso em profundidade PRE-ORDEM	A. ( ) t b z a c d B. ( ) a b t c z d C. ( ) t b a z c d D. ( ) a b c d z t

a.( ) A-1 B-4 C-3 D-2 b.(x) A-2 B-1 C-4 D-3 c.( ) A-1 B-3 C-4 D-2 d.( ) A-3 B-1 C-4 D-2

5.Sobre o algoritmo abaixo é correto afirmar: (1.0 pt)

```
int y(Arv *no, char valor){
    if (no==NULL) return 0;
    else
        if (valor==no->info) return 1;
      else{
          if (no->esq!=NULL){
              int esq=y(no->esq,valor);
              if (esq==0) {
                 if (no->dir!=NULL){
                        int dir=y(no->dir,valor);
                        if (dir==1)
                              return dir;
                       }
                 }
              else
                return esq;
              }
    }
```

a.( ) Faz uma busca por um valor na árvore considerando que ela é uma árvore binária de busca.

b.(X) Faz uma busca por um valor na árvore considerando que ela é uma árvore binária e não está ordenada.

c.( ) Faz a inserção de um valor na árvore considerando que ela é uma árvore binária de busca.

d.( ) Faz a inserção de um valor na árvore considerando que ela é uma árvore binária e não está ordenada.

6. Sobre remoção por cópia e remoção por fusão, assinale a alternativa FALSA: (1.0 pt)

a.(x) Na árvore abaixo, para apagar o nó 50, o algoritmo abaixo usou remoção por fusão:

<< ANTES DE APAGAR O NÓ 50>>	<< DEPOIS DE APAGAR O NÓ 50>>
50 / \ 30 70 / \ / \ 20 40 60 80	60 / \ 30 70 / \ \ 20 40 80

b.( ) Na árvore abaixo, para apagar o nó 50, o algoritmo abaixo usou remoção por cópia:

<< ANTES DE APAGAR O NÓ 50>>	<< DEPOIS DE APAGAR O NÓ 50>>
50 / \ 30 70 / \ / \ 20 40 60 80	60 / \ 30 70 / \ \ 20 40 80

c.( ) A remoção por fusão pode ser ser mais eficiente em termos de operações, pois simplesmente mescla subárvores. Mas em alguns casos pode alterar significativamente a estrutura da árvore, resultando em desbalanceamento.

d.( ) NDA