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

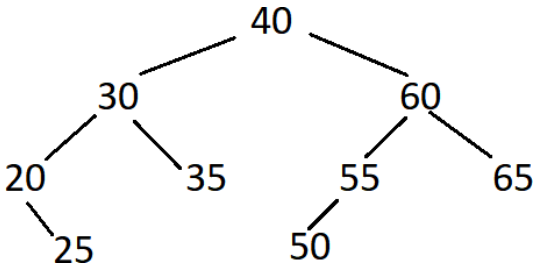
Nome:_____

Prova 1 ED2 – 11.07.2024

GABARITO:
MARQUE AQUI SUAS RESPOSTAS:

QUESTÕES	RESPOSTAS				
1 ***(2 pts)***	**A**	**B**	**C**	**D**	**E**
2 ***(2 pts)***	**A**	**B**	**C**	**D**	**E**
3 ***(1 pt)***	**A**	**B**	**C**	**D**	**E**
4 ***(1 pt)***	**A**	**B**	**C**	**D**	**E**
5.1 ***(1 pt)***	**A**	**B**	**C**	**D**	**E**
5.2 ***(1 pt)***	**A**	**B**	**C**	**D**	**E**
6 ***(2 pts)***	**A**	**B**	**C**	**D**	**E**

1. Sobre a árvore apresentada, associe e marque a alternativa correta: (2.0 pts)



(1) Percurso em Profundidade IN-ORDEM (2) Percurso em Extensão (3) Percurso em Profundidade POS-ORDEM (4) Percurso em Profundidade PRE-ORDEM	A.( ) 20 25 30 35 40 50 55 60 65 B.( ) 25 20 35 30 50 55 65 60 40 C.( ) 40 30 20 25 35 60 55 50 65 D.( ) 40 30 60 20 35 55 65 25 50

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

2. Analise os algoritmos abaixo e associe: (2.0 pts)

(1) Inserção em Árvore Binária (2) Procura de um nó em Árvore Binária

(3) Percurso em Árvore Alinhada (4) Inserção em Árvore Alinhada (5) NDA

(OBS: NEM TODAS ALTERNATIVAS TERÃO CORRESPONDÊNCIA)

```
void metodo1(T el){
        ArvoreNo<T> *p=root,*prev=0;
        while (p!=0){
            prev=p;
            if (el<p->el)
                p=p->left;
            else p=p->right;
        }
        if (root==0)
           root=new ArvoreNo<T>(el);
        else if (el < prev->el)
            prev->left=new ArvoreNo<T>(el);
        else prev->right=new ArvoreNo<T>(el);
}
T metodo2(T el) {
        ArvoreNo<T> *p=root;
        while (p!=0){

            if (el==p->el)
                return p->el;
            else{
                if (el < p->el)
                   p=p->left;
               else p=p->right;
            }
        }
        return 0;
    }
void metodo3(){
        ArvoreNo<T> *prev,*p=root;
        if (p!=0){
            while (p->left!=0)
               p=p->left;
            while (p!=0){
                cout<<p->el<<endl;
                prev=p;
                p=p->right;
                if (p!=0 &&  prev->sucessor==0)
                    while(p->left!=0)
                       p=p->left;
            }
        }
}
```

a. (x) (1) - Metodo1 (2) - Metodo2 (3) - Metodo3

b.( ) (4) - Metodo1 (2) - Metodo2 (5) - Metodo3

c.( ) (4) - Metodo1 (2) - Metodo2 (4) - Metodo3

d.( ) (1) - Metodo1 (2) - Metodo2 (5) - Metodo3

3. 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.

b. (x) Está sendo realizado o percurso em profundidade PRÉ-ORDER. O algoritmo é não-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.

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

4. 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

<C> 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

<D> Uma árvore é balanceada se a diferença na altura de ambas as subárvores de qualquer nó na árvore é maior do que 1 (um).

f

<E> O algoritmo de Morris é aplicado para balancear uma árvore binária, se baseia no fato de que o percurso in-order é muito simples para árvores degeneradas, nas quais nenhum nó tem filhos à esquerda. f

<F> O algoritmo de remoção por fusão usa a estratégia de apagar o nó procurado e também os filhos do nó, caso ele tenha filhos. f

a. ( ) <A>, <C>,<F>

b. ( ) <A>, <E>

c. ( ) <A>, <D>

d. (x) NDA

5. Considere o código abaixo e responda: (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(){
   Arv *a = arvore('a',NULL,NULL);
   Arv *c = arvore('c',0,0);
   Arv *d = arvore('d',0,0);
   Arv *z = arvore('z',c,d);
   Arv *b = arvore('b',a,0);
   Arv *t = arvore('t',b,z);
}
```

5.1 A árvore tem altura igual a:

a.(x) 3 b.( ) 4 c. ( ) 5 d. ( ) nda

5.2. 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

6.Sobre o algoritmo abaixo é correto afirmar: (2.0 pts)

```
int y(Arv *no, char valor){
                    if (no==NUL) 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.