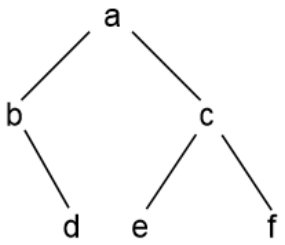
MINISTÉRIO DA EDUCAÇÃO
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ
CURSO : ADS
DISCIPLINA : Estrutura de Dados 2

Nome:_____

Atividade

1. Sobre a árvore apresentada, associe:



( ) percurso em profundidade IN-ORDER ( ) percurso em extensão ( ) percurso em profundidade POS-ORDER ( ) percurso em profundidade PRE-ORDEM	A. ( ) b d a e c f B. ( ) a b d c e f C. ( ) d b e f c a D.( ) a b c d e f

2. Sobre o código abaixo:

```
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);
    }
}
```

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

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

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

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

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

3. Considere o código abaixo e responda:

```
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);
}
```

A árvore tem altura igual a:

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

4. Sobre o código abaixo:

```
class No{
    public:
        char nome;
        No *left;
        No *right;
        No(char n){
            nome=n;
            left=NULL;
            right=NULL;
        }
};
```

```
class Arvore{
    public:
            No *raiz;
            No *pai,*no;
            int h,quantNos;


        Arvore(){
            raiz= NULL;
            pai= NULL;
            no = NULL;
            h = -1;
            quantNos=0;
            }
          void x(No *arv,No *n){
              if (isEmpty()==1)
                    raiz=n;
               else{
                    No *percorre=raiz;
                    while (percorre!=NULL){
                            if (percorre->nome<n->nome)
                               if (percorre->right==NULL){
                                    percorre->right=n;
                                    break;
                               }
                               else
                                   percorre=percorre->right;
                            if (percorre->nome>n->nome)
                               if (percorre->left==NULL){
                                    percorre->left=n;
                                    break;
                               }
                            else
                               percorre=percorre->left;
                    }
             }
        } }

        void imprime(No *arv){
                    if (arv!=NULL){
                            cout<<"<"<<arv->nome;
                            imprime(arv->left);
                            imprime(arv->right);
                            cout<<">";
                    }
                    else
                       cout<<"<>";

            }    }
```

É correto afirmar que:

a.( ) O método “x” é recursivo e insere um novo nó em uma árvore binária de busca.

b.( ) O método “x” não é recursivo e faz a busca por um nó em uma árvore binária de busca, verificando se o nó existe ou não na árvore.

c.( ) O método “x” é recursivo e faz a busca por um nó em uma árvore binária de busca, verificando se o nó existe ou não na árvore.

d.( ) O método “x” não é recursivo e insere um novo nó em uma árvore binária de busca.

5. Ainda sobre o código da questão 4, analise os trechos abaixo e marque a afirmação correta:

(a)

```
main(){
    Arvore *arvore=new Arvore();
    No *w = new No('W');
    No *z = new No('Z');
    No *a= new No('A');
    No *b = new No('B');
    arvore->x(arvore->raiz,w);
    arvore->x(arvore->raiz,b);
    arvore->x(arvore->raiz,z);
    arvore->x(arvore->raiz,a);
    arvore->imprime(arvore->raiz);
}
```

(b)

```
main(){
    Arvore *arvore=new Arvore();
    arvore->x(arvore->raiz,’w’);
    arvore->x(arvore->raiz,’b’);
    arvore->x(arvore->raiz,’z’);
    arvore->x(arvore->raiz,’a’);
    arvore->imprime(arvore->raiz);
}
```

(C)

```
main(){
    Arvore *arvore=new Arvore();
    No *w = new No('W');
    No *z = new No('Z');
    No *a= new No('A');
    No *b = new No('B');
    arvore->x(arvore,w);
    arvore->x(arvore,b);
    arvore->x(arvore,z);
    arvore->x(arvore,a);
    arvore->imprime(arvore);
}
```

a.( ) Somente o código (A) está correto, e será impresso a árvore <W<B<A<><><>><><>><Z<><>>>

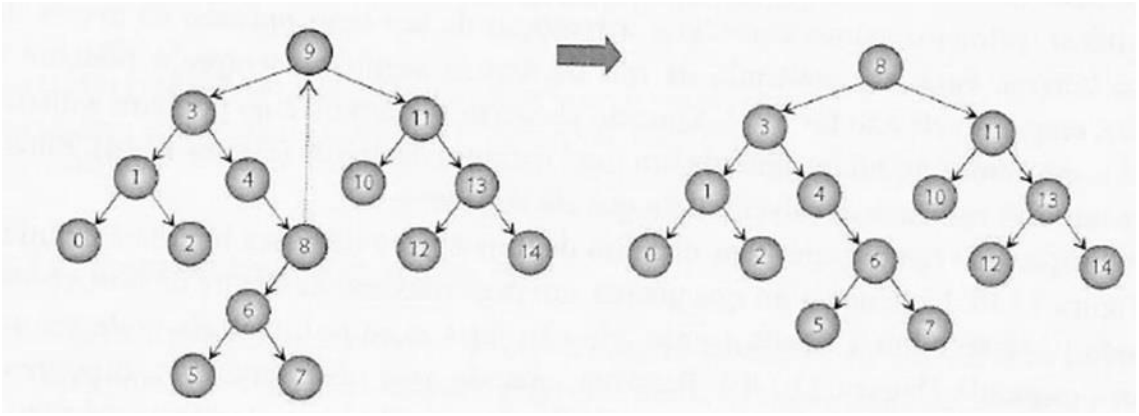
b.( ) Somente o código (A) está correto, e será impresso a árvore <W<Z<A<><><>><><>><B<><>>>

c.( ) Os códigos (A), (B) e (C) estão corretos, e em todos eles será impresso a árvore <W<B<A<><><>><><>><Z<><>>>

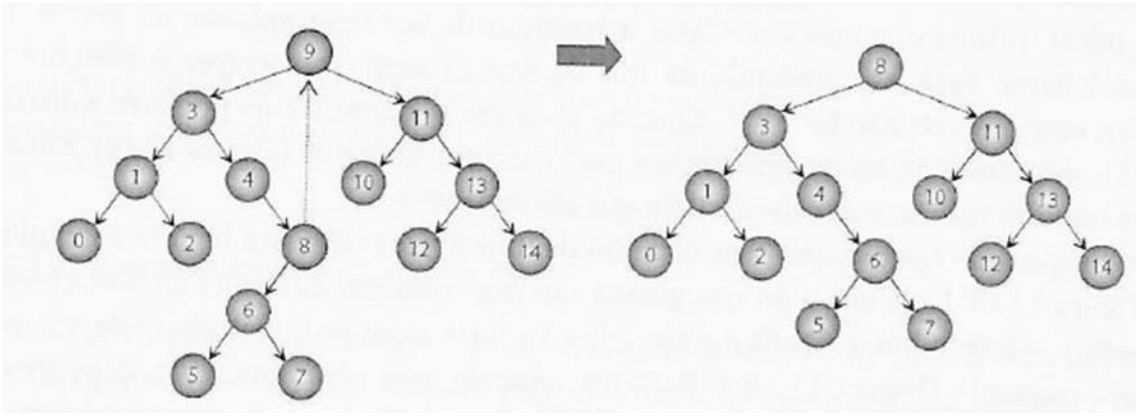
d.( ) NDA

6. Sobre a remoção em árvore, marque a(s) alternativa (s) VERDADEIRA (S). OBS: Leia com atenção as alternativas, sua resposta deve conter TODAS as opções verdadeiras para que possa ser considerada correta.

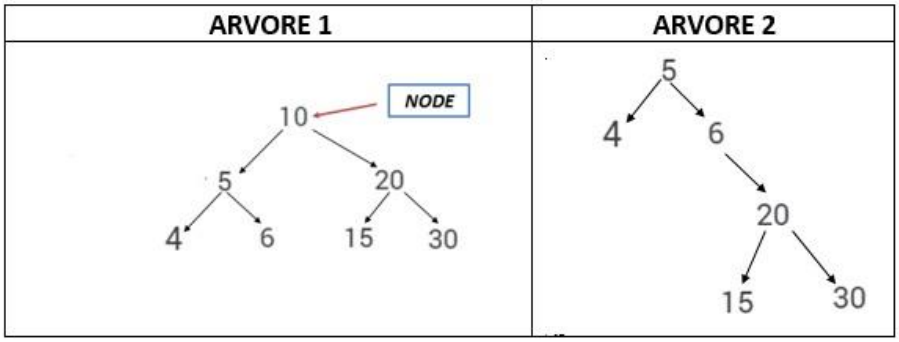
a.( ) A figura mostra o uso da remoção por cópia. Nesse caso o nó que deve ser apagado (no exemplo abaixo o nó a ser removido é o nó "9") não é removido na memória, apenas acontece a substituição do valor armazenado nele (na figura abaixo, ele é substituido pelo nó "8").



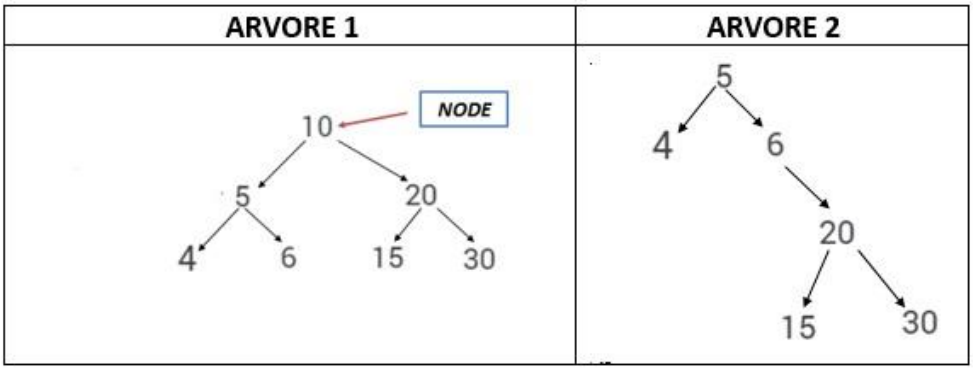
b. ( ) A figura mostra o uso da remoção por fusão. Nesse caso o nó que deve ser apagado (no exemplo abaixo o nó a ser removido é o nó "9") não é removido na memória, apenas acontece a substituição do valor armazenado nele (na figura abaixo, ele é substituido pelo nó "8").



c. ( ) Para apagar o elemento NODE=10 apresentado na ÁRVORE 1, aplicando-se o algoritmo de REMOÇÃO POR FUSÃO, o resultado será a árvore apresentada em ÁRVORE 2.



d. ( ) Para apagar o elemento NODE=10 apresentado na ÁRVORE 1, aplicando-se o algoritmo de REMOÇÃO POR CÓPIA, o resultado será a árvore apresentada em ÁRVORE 2.



7. Leia as afirmações abaixo e associe:

(1) O último a entrar é o primeiro a sair.
(2) O último a entrar é o último a sair.
(3) Para cada nó ***n*** da árvore, todos os valores armazenados em sua subárvore à esquerda são menores que o valor ***v*** armazenado em ***n.*** E todos os valores armazenados na subárvore à direita são maiores que ***v***.
(4) São estruturadas de forma hierárquica e a distância entre cada nó e a raiz é denominada de ***nível***.
(5) Estrutura de dados onde o número de filhos é indefinido.
(6) Estrutura de dados onde, cada nó, tem no máximo 2 (dois) filhos.

( ) Árvore genérica

( ) Pilha

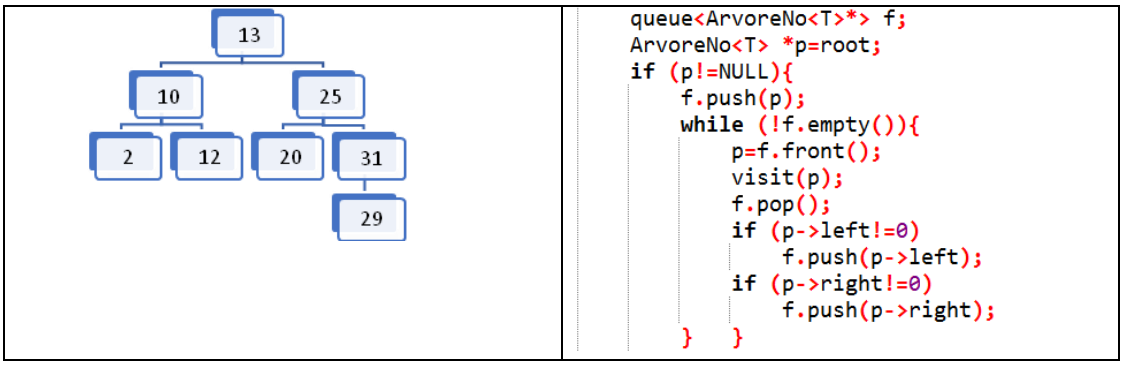
( ) Árvore Binária

( ) Fila

( ) Árvore Binária de Busca

( ) Árvore

8.Considerando a árvore binária apresentada e aplicando o código ao lado nesta árvore, o percurso resultante será:




```
queue<ArvoreNo<T>*> f;
ArvoreNo<T> *p=root;
if (p!=NULL){
    f.push(p);
    while (!f.empty()){
        p=f.front();
        visit(p);
        f.pop();
        if (p->left!=0)
            f.push(p->left);
        if (p->right!=0)
            f.push(p->right);
    }   }
```

a.( ) 13 10 25 2 12 20 31 29

b.( ) 29 2 12 20 31 10 25 13

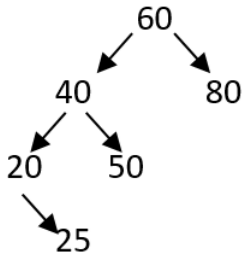
c.( ) 2 10 12 13 20 25 29 31

d.( ) NDA

9. Considerando o algoritmo:

```
void [redacted] {
    ArvoreNo<T> *p=root, *tmp;
    while (p!=0){
        if (p->left==0){
            visit(p);
            p=p->right;
        }
        else {
            tmp = p->left;
            while (tmp->right!=0 && tmp->right!=p)
                tmp=tmp->right;
            if (tmp->right==0){
                tmp->right=p;
                p=p->left;
            }
            else{
                visit(p);
                tmp->right=0;
                p=p->right;
            }
        }
    }//fim while
}
```

E a árvore:



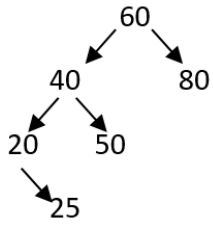
a) O que o algoritmo está fazendo?

b) Qual resultado apresentado pelo algoritmo?

10. Considerando o algoritmo:

```
void [redacted](){
    ArvoreNo<T> *p=root, *tmp;
    while (p!=0){
        if (p->left==0){
            visit(p);
            p=p->right;
        }
        else{
            tmp = p->left;
            while (tmp->right!=0 && tmp->right!=p)
                tmp=tmp->right;
                if (tmp->right==0){
                    visit(p);
                    tmp->right=p;
                    p=p->left;
                }
                else{
                    tmp->right=0;
                    p=p->right;
                } /*fim else*/  }
        } /*fim while*/ }
```

E a árvore:



a) O que o algoritmo está fazendo?

b) Qual resultado apresentado pelo algoritmo?