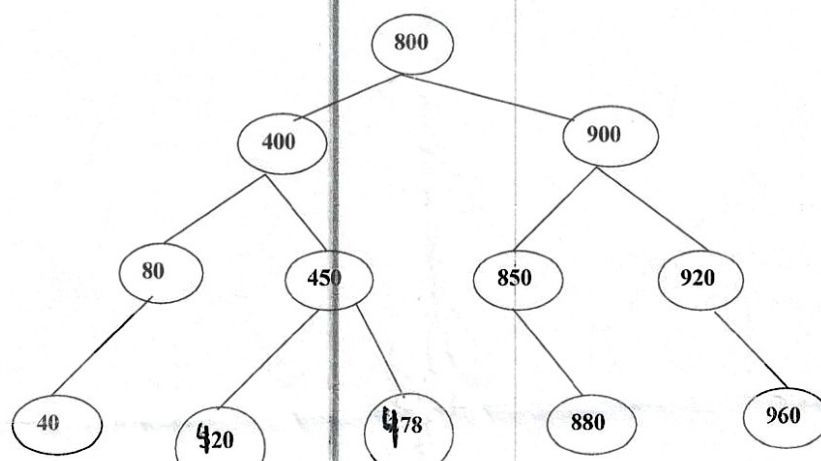


UFMA – CCET - DEINF
Estrutura de Dados
Terceira Avaliação - C

1. Considere a árvore binária de pesquisa da figura abaixo:

- Considerando que a visitação imprime o valor de cada nó, escreva resultado da visitação da árvore em pré-ordem, pós-ordem e ordem simétrica.
- Faça a inserção dos dados (consecutivamente) a seguir (mostre o resultado de cada passo desenhando a árvore):
a) 846 b) 867 c) 34
- Remova os seguintes nós consecutivamente
a) 900 b) 80 c) 960



2. Escreva um algoritmo que recebe duas árvores topologicamente iguais e calcula o número de nós correspondentes que tem o mesmo valor nas duas árvores, use obrigatoriamente o protótipo do algoritmo abaixo.

`int abDiferencaNósCorresp (TNode *t1, TNode *t2, int (*getvalue) (void *))`

OBS: a função `getvalue` retorna o valor inteiro que está armazenado no nó da árvore.

3. Escreva um algoritmo para calcular o número de nós de uma árvore binária que possuem altura menor que `h`.

`int abNumNosMenorQueH (TNode *t, int h);`

int level.

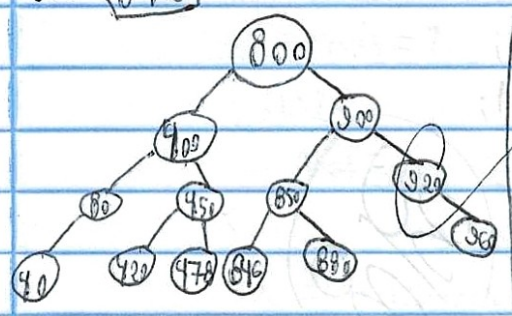
4. Escreva um algoritmo que recebe uma árvore binária e retorna o número de nós de grau 1 na árvore.

`int abCalcNumNosGrau1(TNode *t)`

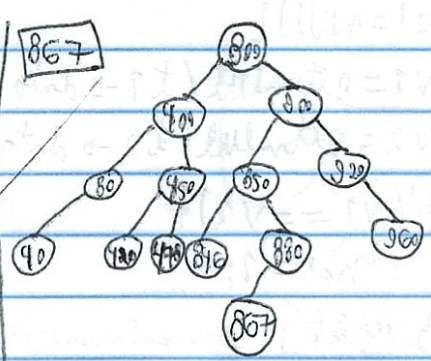
100
100

1) a. pre-ordm: 800, 400, 80, 40, 450, 420, 478, 900, 850, 880, 920, 960
 simétrica: 40, 80, 400, 420, 450, 478, 800, 850, 880, 900, 920, 960
 pos-ordm: 40, 80, 420, 478, 450, 400, 880, 850, 960, 920, 900, 800

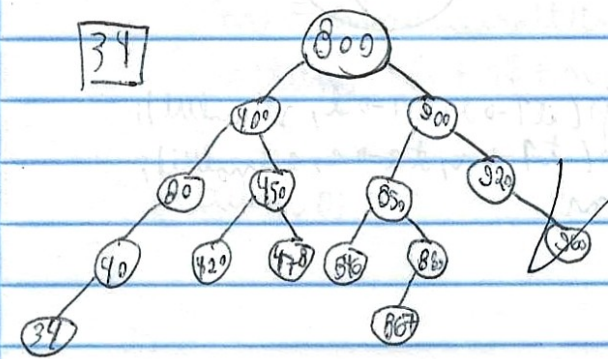
b. 896



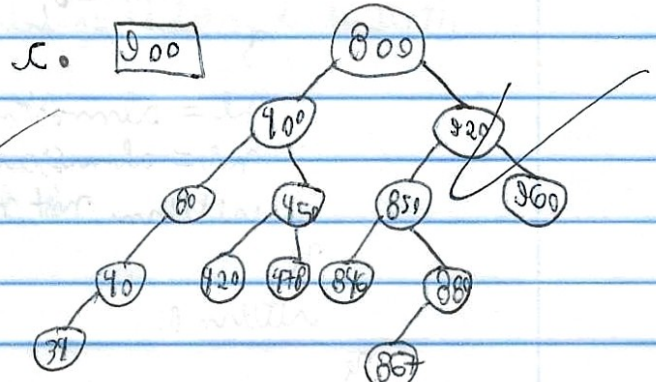
867



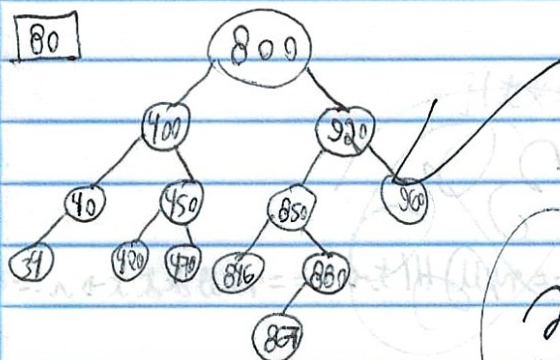
34



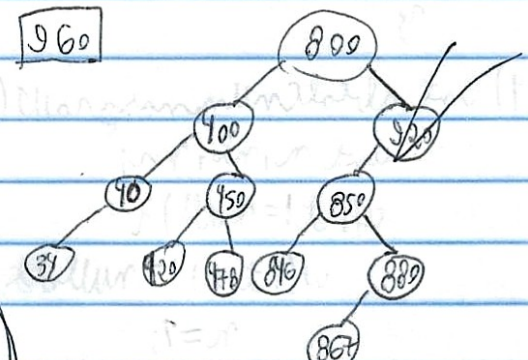
c. 900



80



960



30
30


```

2) int abnosszerp(t_nodl * t1, t_nodl * t2, int(x getnolul / mid *)) {
    int r1, r2, n1, n2, n;
    if (t1 != null) {
        if (t2 != null) {
            v1 = getnolul(t1 -> data);
            v2 = getnolul(t2 -> data);
            if (v1 == v2) {
                n1 = 1;
            } else {
                n1 = 0;
            }
            n1 = abnosszerp(t1 -> l, t2 -> l, getnolul);
            n2 = abnosszerp(t1 -> r, t2 -> r, getnolul);
            return n1 + n2 + n;
        } else {
            return 0;
        }
    } else {
        return 0;
    }
}

```

30/30

```

4) int abcsolnomszall(t_nodl * t) {
    int n, n1, n2;
    if (t != null) {
        if ((t -> l != null && t -> r == null) || (t -> l == null && t -> r != null)) {
            n = 1;
        } else {
            n = 0;
        }
        n1 = abcsolnomszall(t -> l);
        n2 = abcsolnomszall(t -> r);
        return n + n1 + n2;
    } else {
        return 0;
    }
}

```

30/30

3) int abnussalttkamenquell(tnode* t, int d, int level){

int nl, m, nos;

if(t != null){

if(level > d){

nos = 1;

} else{

nos = 0;

}

nl = abnussalttkamenquell(t->l, d, level+1);

m = abnussalttkamenquell(t->r, d, level+1);

return nos + nl + m;

}

return 0;

}

30
30