1. Percorra a árvore em pré-ordem, em-ordem e pos-ordem.

a) Pré-ordem:

R:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

b) Em-ordem:

R:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

c) Pós-ordem:

R:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

10

20

50

45

3

6

9

5

4

1. A partir do código para imprimir uma árvore binária de busca, de maneira recursiva, escreva o código para imprimir de maneira decrescente).

void AVL\_tree::printrecursive(AVL\_node \*r)

{

if(r!=NULL){

cout<<r->key;

printrecursive(r->left);

printrecursive(r->right);

}

}

1. Construa uma árvore AVL, de maneira que ela sempre fique balanceada (isto é, se a inserção de um elemento causar o desbalanceamento da árvore, é necessário balanceá-la novamente). Apresente a árvore passo-a-passo. Os dados são (nesta ordem): 50, 40, 30, 13, 12, 11
2. Dada a árvore do exercício 1, remova o elemento 6.
3. Considere uma árvore de busca binária e uma árvore AVL de altura 4. Mostre como seria a estrutura destas duas árvores com o número mínimo de nós e com o número máximo de nós.
4. Mostre os resultados de inserir as chaves a seguir em uma árvore B de ordem 2 inicialmente vazia: 14, 39, 1, 6, 41, 32, 8, 38, 43, 3, 36. Após essas inserções, mostre os resultados de remover as chaves ímpares nessa estrutura.