Tópico: **Árvore Binária**

1. Desenhe a árvore binária resultante da inserção de 25, 11, 9, 33, 17, 5, 45, 50, 27, 30, 22, 12 e 8, nessa ordem em uma árvore inicialmente vazia.

2. Desenha a árvore binária obtida a partir da árvore do exercício anterior, fazendo a remoção das chaves 12, 11 e 25, nessa ordem.

3. Uma implementação possível para árvores binárias é utilizando um vetor alocado estaticamente que comporta no máximo n nodos, tal que a raiz está sempre localizada na posição 0. Além disso, dado um nó da posição i, o filho à esquerda está na posição $2i+1$ e o filho à direita na posição $2i+2$. Considerando essa implementação, caracterize a situação em que o vetor está completamente preenchido. Também calcule a altura máxima que a árvore pode ter considerando um vetor com n posições.

4. Considere o código abaixo:

```
struct no {  // estrutura de nó para árvore binária
   int info;
   struct no* esq;
   struct no* dir;
};

typedef struct no* arvore;

int vazia(arvore r){
      return (r == NULL);
   }
```

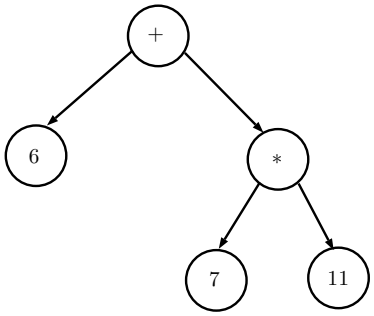
   Implemente as seguintes funções, tanto de maneira recursiva quanto iterativa:

   (a) *inserir* que insere uma chave dada como entrada.
   (b) *remover* que remove uma chave dada como entrada.
   (c) *in_ordem* que imprime as chaves fazendo o percorrimento *in ordem*.
   (d) *pre_ordem* que imprime as chaves fazendo o percorrimento *pré-ordem*.
   (e) *pos_ordem* que imprime as chaves fazendo o percorrimento *pós-ordem*.
   (f) *maximo* que retorna o valor da maior chave.
   (g) *minimo* que retorna o valor da menor chave.
   (h) *altura* que retorna a altura da árvore binária.
   (i) *soma* que retorna a soma de todas as chaves.

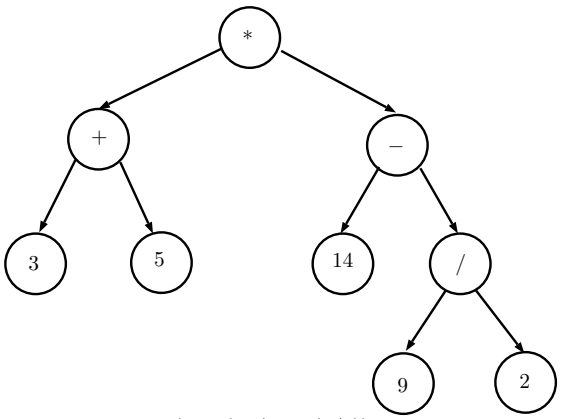
5. Considerando o código do exercício anterior, implemente a função *imprimir_por_niveis* que imprime as chaves por níveis.

6. Implemente um algoritmo para determinar se uma árvore binária é cheia.

7. Implemente um algoritmo para determinar se uma árvore binária é estritamente binária.



$6 + 7 * 11$



$(3 + 5) * (14 - (9/2))$

8. Árvores binárias podem ser usadas para representar expressões aritméticas como acima:

   (a) Implemente uma árvore binária para representar expressões aritméticas como as do exemplo anterior

   (b) Implemente uma função *calcular* que recebe uma árvore binária representando uma expressão aritmética e que retorna o seu valor.

   (c) Implemente uma função *constroi_arvore* que recebe uma string contendo expressões aritméticas formanda por operandos inteiros, $+,-,*,/$, (,), e constrói a árvore binária correspondente. Observe que os parêntesis não são representados explicitamente na árvore.

   (d) Implemente uma função $prefixa$ que retorna uma string contendo a notação prefixa correspondente à expressão aritmética.

   (e) Implemente uma função $infixa$ que retorna uma string contendo a notação infixa correspondente à expressão aritmética.

   (f) Implemente uma função $posfixa$ que retorna uma string contendo a notação posfixa correspondente à expressão aritmética.