Tópico: **Árvores 2-3, B, B* e B$^+$**

1. Desenhe a árvore 2-3 resultante da inserção da chaves 5, 7, 2, 11, 50, 35, 22, 13, 37, 6, 9, 44, 19, 83, 12, 8 e 21, nessa ordem em uma árvore inicialmente vazia.

2. A partir da árvore obtida no exercício anterior, desenhe a árvores 2-3 resultante da remoção das chaves 21, 22 e 11 , nessa ordem.

3. Considere o código abaixo:

```
struct no23 {
     int chave_esq;      // chave esquerda
     int chave_dir;      // chave direita
     struct no23 * esq;  // subárvore esquerda
     struct no23 * meio; // subárvore do meio
     struct no23 * dir;  // subárvore direita
     int n;              // número de chaves no nó
  };

typedef struct no23* arvore23; //árvore é um ponteiro
                               //  para um nó
```

   Implemente funções para inserção, pesquisa, e remoção de chaves na árvore 2-3.

4. Implemente uma função `int minimo(arvore23 r)` que retorna o menor valor de chave presente em uma árvore 2-3.

5. Implemente uma função `int maximo(arvore23 r)` que retorna o maior valor de chave presente em uma árvore 2-3.

6. Implemente uma função `conta_nos(arvore23 r)` que retorna o número de nós em uma árvore 2-3.

7. Implemente uma função `in_ordem(arvore23 r)` que imprima as chaves de uma árvore 2-3 em ordem crescente.

8. Considere o código abaixo:

```
struct no23 {
     int chave_esq;      // chave esquerda
     int chave_dir;      // chave direita
     struct no23 * esq;  // subárvore esquerda
     struct no23 * meio; // subárvore do meio
     struct no23 * dir;  // subárvore direita
     struct no23 * pai;  // ponteiro para o pai
     int n;              // número de chaves no nó
  };

typedef struct no23* arvore23; //árvore é um ponteiro
                               //  para um nó
```

   Implemente as funções inserção e remoção tal que cada nó tenha um ponteiro para o nó-pai. No caso da raiz, o pai é `NULL`.

9. Considere o código abaixo:

```
//estrutura de nó para árvore B
//tem uma posição a mais de chave e ponteiro de filho para
//facilitar a implementação da operação split
typedef struct no {
   int numChaves;
   int chave[ORDEM];
   struct no* filho[ORDEM+1];
} arvoreB;
```

   Implemente funções para inserção, busca e remoção de chaves em uma árvore B.

10. Por que a redistribuição (empréstimo) de chaves durante a remoção em árvore B deve ser tentada primeiro antes da concatenação?

11. Considere uma árvore B de ordem 3 e altura 4. Quantos nós tem essa árvore?

12. Implemente função `int maximo(arvoreB* r)` que retorna a maior chave presente na árvore B.

13. Implemente função `int minimo(arvoreB* r)` que retorna a menor chave presente na árvore B.

14. Implemente função `int conta_nos_minimo_chaves(arvoreB* r)` que retorna o número de nós cujo número de chaves seja mínimo de acordo com a ordem da árvore.

15. Considere as chaves 22, 3, 9, 44, 5, 11, 8, 25, 33, 67, 6, 17, 99, 7, 56, 80, 19, 55, 1, 4, 36, 27 e 13, nessa ordem.

    (a) Desenhe a árvore B de ordem 3 resultante da inserção das chaves.

    (b) Desenhe a árvore B de ordem 5 resultante da inserção das chaves.

    (c) Desenhe a árvore B de ordem 7 resultante da inserção das chaves.

    (d) Desenhe a árvore B* de ordem 5 resultante da inserção das chaves.

    (e) Desenhe a árvore B$^+$ de ordem 5 resultante da inserção das chaves.

16. Para cada uma das árvore obtidas no exercício anterior, desenhe a árvore resultante da remoção das chaves 22, 19, 17 e 13, nessa ordem.

17. (**Desafio!**) Considere o código abaixo:

```
#define ORDEM 5
 //estrutura de nó para árvore B+
typedef struct nodeBMais {
    void * ponteiro[ORDEM]; // vetor de ponteiros
    int chave[ORDEM-1];     // vetor de chaves
    struct nodeBMais * pai; // ponteiro para o nó-pai
    int eh_folha;           // booleano, verdadeiro quando nó é folha
    int numChaves;          // número de chaves no nó
} noBMais;
```

   Observações:

   - `ponteiro` é um vetor de ponteiros do tipo `void`, podendo apontar tanto para subárvores quanto para dados satélites relacionados às chaves.
   - `chave` é um vetor de inteiros para armazenar as chaves.
   - `pai` é um ponteiro para o nó-pai visando facilitar as operações de inserção e remoção.
   - se `eh_folha` é verdadeiro então `ponteiro[ORDEM]` guarda o endereço do nó-folha à direita.

   Implemente funções para inserção, busca, remoção e percorrimento in-ordem na árvore B$^+$.