Nome: João Ricardo Malta De Oliveira
Matrícula: 2112714
Nome: Vinicius Machado Da Rocha Viana
Matrícula: 2111343

Estrutura da Árvore B:

A árvore B é uma estrutura de dados que é usada para armazenar chaves ordenadas e permite uma recuperação eficiente dessas chaves. A árvore B possui uma ordem que determina o número máximo e mínimo de elementos permitidos em cada nó. A struct que utilizamos para construção árvore segue a seguinte estrutura:

```
typedef struct no Page;
struct no
{
  int qtdElem;
  int chave[MAX];
  Page *ramo[MAX + 1];
};
```

Os campos construídos na struct foram utilizados para: qtdElem armazenar a quantidade de elementos presentes no nó, chave[MAX] é um array para armazenar as chaves do vetor e por fim, ramo[MAX + 1] que é um array de ponteiros para outros nós da árvore.

Função cria:

A função cria é responsável por criar um novo nó da árvore B e inicializá-lo com os elementos e ramos fornecidos, ela recebe como parâmetro a quantidade de elementos presentes no vetor a ser criado, as chaves que estarão nesse vetor e o ponteiro para o filho. A lógica que utilizamos para sua construção foi a seguinte:

```
Page *cria(int qtdElem, int chaves[], Page *filhos[]);
```

A função começa alocando memória para um novo nó da árvore B. Se a alocação falhar, ela imprime uma mensagem de erro e retorna NULL, se não o novo nó é inicializado com a quantidade de elementos qtdElem fornecida como argumento. Em seguida, copiamos as chaves fornecidas no array chaves[ ] para o novo nó, através de um for que percorre cada chave e a cópia para o campo chave do nó. O próximo passo foi iniciar os ponteiros para os nós filhos no array ramo[ ] do novo nó. Isso é feito com outro for que percorre os elementos do array filhos[ ] e atribui os ponteiros aos ramos correspondentes do novo nó e para finalizar a função retornamos o novo nó.

Função intervalo:

A função intervalo recebe um ponteiro para um nó da árvore B, ***arv***, juntamente com dois valores inteiros, ***lim_inf*** e ***lim_sup***, indicando o intervalo desejado. O algoritmo então percorre a árvore em ordem simétrica.
A função é implementada de maneira recursiva. Inicialmente, verifica se o nó fornecido não é nulo. depois, percorre os nós filhos à esquerda do nó atual, chamando recursivamente a função para cada nó filho à esquerda, se ele for diferente de nulo. Após esse passo, a função imprime as chaves do nó atual se estiverem dentro do intervalo definido.. Por fim, continua percorrendo os filhos à direita do nó atual, novamente chamando recursivamente a função para cada nó filho à direita, se eles forem diferentes de nulo.
A função continua até que todos os elementos dentro do intervalo tenham sido impressos.

Função Main:

Na função main criamos uma estrutura de árvore B de ordem 2 e chamamos a função intervalo para imprimir chaves dentro de um intervalo especificado. Declaramos as variáveis:
Page* nulos[MAX + 1]: Array de ponteiros para nós nulos.
int arr1[ ]: Array de chaves.
int arr2[ ]: Array de chaves.
…
int arr6[ ]: Array de chaves

Após isso criamos 9 nós da árvore B (raiz1, raiz2, raiz3, ...) com diferentes chaves e ramos, tais nós são preenchidos com as chaves e ramos correspondentes usando a função cria. Também definimos arrays de nós primários (folhas1, folhas2) que contém os nós criados anteriormente, nós primários (no1 e no2) com chaves específicas e ramos associados e o nó principal head. A função intervalo é chamada com o nó principal head e os limites do intervalo [10, 250]. Isso resulta na impressão das chaves que estão dentro do intervalo especificado.