Implemente o método de ordenação interna HeapSort. Responda, usando um arquivo PDF, como fez a atividade, quais problemas teve e como os superou, colocando prints do código e do resultado para ilustrar o que foi construído.

```
//FUNÇÃO PARA CONSTRUIR A HEAP
void constroi(vector<int> &arr, int n, int i) {
    int maior = i;
    int esq = 2 * i + 1;
    int dir = 2 * i + 2;

    //ENCONTRE ÍNDICE DO MAIOR ELEMENTO ENTRE PAI E OS FILHOS
    if (esq < n && arr[esq] > arr[maior])
        maior = esq;

    if (dir < n && arr[dir] > arr[maior])
        maior = dir;

    //SE MAIOR ELEMENTO NÃO FOR O PAI, TROCA E CONTINUA A CONSTRUÇÃO
    if (maior != i) {
        swap(arr[i], arr[maior]);
        constroi(arr, n, maior);
    }
}
```

A função **constroi** é usada para construir um **heap** a partir de um vetor. Ela toma o vetor, seu tamanho **n** e um índice **i** como argumentos. Essa função será chamada recursivamente para garantir que a propriedade de max-heap seja mantida.

```
//FUNÇÃO PARA REFAZER A HEAP DE CIMA PARA BAIXO
void refazDeCimaParaBaixo(vector<int> &arr, int n) {
    //CONSTRUÇÃO COMEÇA A PARTIR DO ÚLTIMO PAI
    for (int i = n / 2 - 1; i >= 0; i--)
        constroi(arr, n, i);
}
```

Reorganiza a heap de cima para baixo, começando dos **nós pai** e descendo até os **nós folha**, garantindo que a propriedade de max-heap seja mantida.

```
//REFAZ A HEAP DE BAIXO PARA CIMA APÓS INCERSÃO
void refazDeBaixoParaCima(vector<int> &arr, int i) {
    while (i > 0 && arr[(i - 1) / 2] < arr[i]) {
        swap(arr[i], arr[(i - 1) / 2]);
        i = (i - 1) / 2;
    }
}
```

Refaz a heap de baixo para cima após a inserção de um novo elemento, garantindo novamente que a propriedade de max-heap seja mantida.

```cpp
//FUNÇÃO PARA REMOVER O ELEMENTO MÁXIMO DA HEAP
void remove(vector<int> &arr, int &n) {
    if (n <= 0)
        return;

    //TROCA O ELEMENTO MÁXIMO COM O ÚLTIMO ELEMENTO
    swap(arr[0], arr[n - 1]);
    n--;

    //RECONSTRÓI A HEAP APÓS A REMOÇÃO
    constroi(arr, n, 0);
}
```

Remove o elemento máximo **(raiz)** da heap. Ela troca o elemento máximo com o último elemento do vetor, diminui o tamanho da heap e, em seguida, reconstrói a partir do elemento trocado.

```cpp
//FUNÇÃO HEAP SORT
void heapSort(vector<int> &arr) {
    int n = arr.size();

    //CONSTRÓI A HEAP INICIAL
    refazDeCimaParaBaixo(arr, n);

    //EXTRAI ELEMENTOS UM POR UM E RECONSTRÓI A HEAP
    for (int i = n - 1; i > 0; i--) {
        swap(arr[0], arr[i]);
        constroi(arr, i, 0);
    }
}
```

Essa é a função principal de ordenação usando o Heap Sort. Ela **constrói** a heap inicial e, em seguida, extrai os elementos um por um da heap e **reconstrói** a heap para obter a ordenação desejada.

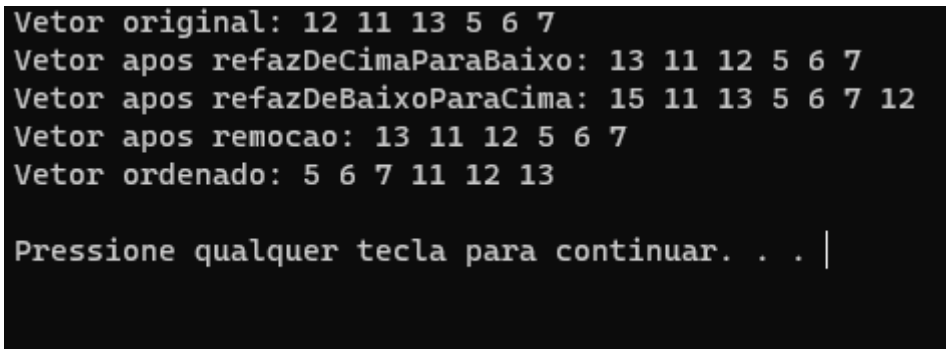
```cpp
int main() {
    vector<int> arr = {12, 11, 13, 5, 6, 7};
    int n = arr.size();

    cout << "Vetor original: ";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;

    //APLICA A FUNÇÃO REFAZDECIMAPARABAIXO PARA CONSTRUIR A HEAP
    refazDeCimaParaBaixo(arr, n);

    cout << "Vetor apos refazDeCimaParaBaixo: ";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;

    //INSERE UM NOVO ELEMENTO E RECONSTRÓI A HEAP DE BAIXO PARA CIMA
    arr.push_back(15);
    n++;
    refazDeBaixoParaCima(arr, n - 1);

    cout << "Vetor apos refazDeBaixoParaCima: ";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;
```

```cpp
    //REMOVE O ELEMENTO MÁXIMO (RAIZ DA HEAP)
    remove(arr, n);

    cout << "Vetor apos remocao: ";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;

    //APLICA O ALGORITMO HEAP SORT PARA ORDENAÇÃO
    heapSort(arr);

    cout << "Vetor ordenado: ";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;

    return 0;
}
```

Aqui começa a função principal do programa. Um vetor é inicializado com alguns valores. O programa imprime o vetor original e, em seguida, aplica as diferentes funções definidas acima para demonstrar a construção da heap, a inserção de um novo elemento, a remoção do elemento máximo e, finalmente, a ordenação do vetor usando o Heap Sort.

Cada seção dentro da função main() demonstra uma etapa diferente do algoritmo Heap Sort e como as funções auxiliares contribuem para construir e manipular a heap. No final, o vetor é ordenado e impresso em ordem crescente.

RESULTADO:



COMENTÁRIO: a construção desse código foi um pouco tranquila graças ao material que o professor disponibilizou, nele já continha as principais funções necessárias para a execução de tal. Minha maior dificuldade foi de início ao tentar entender o que era para ser feito, mas depois de uma boa explicação foi bem tranquilo.