3. Ternary Heap Analysis

Briefly analyze the running time for your heap's insert( ) and removeMin( ) operation.

insert( ) analysis:

```
insert( arr, key ){
        arr.push_back(key);                     - constant time
        percolate_up(arr.size( )-1,arr);        - this operation is O(log n)
}
```

Thus, insert( ) takes O(log n) time

removeMin( ) analysis:

```
removeMin(arr){
        int root = arr.front( );                - constant time
        arr.front( ) = arr[arr.size( )-1];      - constant time
        arr.pop_back( );                        - constant time
        min_heapify(0, arr);                    - this operation is O(log n)
        return root;                            - constant time
}
```

Thus, removeMin( ) takes O(log n) time

As part of your analysis, answer the following questions for each operation

(a) Is it asymptotically faster or slower than the same operation in a binary heap?
Ans: Asymptotically these operations of ternary heap are as same as those of binary heap. Because, percolateUp( ) and minHeapify( ) operations just compare the parent node with its children and selects one of them. This doesn't change asymptotically just because you've to take minimum of 3 children instead of 2. The asymptotic running time is the same.

(b) Would you expect it to have a larger or a smaller constant factor than the same operation with a binary heap?
Ans: The operations have asymptotically same running time, but ternary heap operations would have a larger constant factor than binary heap operations. Ternary heap has nodes each with 3 children as compared to binary heap with nodes having only 2 children. It will take more time for ternary heap operations to compare the values of all the children and then come up with the minimum value.