Assumptions for Array Implementation of Max Heap ADT

Max Heap keeps the largest value at the root node of the binary tree.
Instance variables

myData - fixed sized array of element type (int for this example)
myCount - integer count of current number of elements stored
Initial value of myCount is 0 to indicate an empty heap.

Insert (X)
Given an integer element X to insert. Into what array index slot do we initially insert it in order to retain the heap shape? Insert into next open slot.

```
int startingIndex = myCount;

Percolate new element X upwards until it reaches
its correct place.

int hole = startingIndex ; // Array index where X
starts.

// (1) keep going as long as hole index is?
while( hole > 0 &&

// (2) and new element is > than value of parent
       X > myData[ (hole-1)/2 ] ))
{
     // (3) Parent's value drops down into hole
     myData[ hole ] = myData[ (hole-1)/2 ];

     // (4) Change hole index to that of parent
     hole = (hole-1)/2;

}// END While
array[ hole ] = X;
myCount = myCount + 1;
```

Remove root element from Max Heap
Of course the heap cannot be empty, to remove.

```
// Where does the max value always reside?
int MaxValue = myData[ 0 ];

// What value will take the root?  Last one.

myData[ 0 ] = myData[ mySize-1 ];
mySize--;

// Begin percolate down at index of root
int hole = 0;

int child;
int temp = myData[ hole ];

while( hole * 2 + 1 < mySize)
{
   // Index of left child of node in hole index

   child = 2 * hole + 1;

   if( child != mySize &&
       myData[ child + 1 ] > myData[ child ] )
           child++;

   if( myData[ child ] > temp )
      myData [ hole ] = myData[ child ];
   else
      break;

   hole = child;
}
myData[ hole ] = temp;
```