// assumes that array is called 'elements'

```cpp
template <class Comparable>
void BinaryHeap<Comparable>::insert(const Comparable & x)
{
    if (isFull()) throw Overflow();

    // Percolate up
    int hole = ++currentSize;
    for (; hole > 1 && x < array[ hole / 2 ]; hole /= 2 )
        array[ hole ] = array[ hole / 2 ];
    array[ hole ] = x;
}
```

```cpp
template <class Comparable>
void BinaryHeap<Comparable>::deleteMin( Comparable & minItem)
{
    if (isEmpty()) throw Underflow();

    minItem = array[ 1 ];
    array[ 1 ] = array[ currentSize-- ];
    trickleDown( 1 );
}
```

```cpp
template <class Comparable>
void BinaryHeap<Comparable>::trickleDown(int hole)
{
    int child;
    Comparable tmp = array[ hole ];

    for( ; hole * 2 <= currentSize; hole = child )
    {
        child = hole * 2;
        if(child != currentSize && array[child + 1] < array[child])
            child++;
        if (array[child] < tmp) array[hole] = array[child];
        else break;
    }
    array[ hole ] = tmp;
}
```