

SORT

I. Java Implementation

insertionSort

```
@Assertion(value = { @Case(
pre = "a !=null",
post = "isSorted(a)") })
public void insertionSort( int[ ] a )
{
    int j;
    for( int p = 1; p < a.length; p++ )
    {
        int tmp = a[ p ];
        for( j = p; j > 0 && tmp < a[ j - 1 ]; j-- )
            a[ j ] = a[ j - 1 ];
        a[ j ] = tmp;
    }
}
```

shellSort

```
@Assertion(value = { @Case(
pre = "a !=null",
post = "isSorted(a)") })
public void shellsort(int[ ] a )
{
    int j;
    for( int gap = a.length / 2; gap > 0; gap /= 2 )
        for( int i = gap; i < a.length; i++ )
        {
            int tmp = a[ i ];
            for( j = i; j >= gap && tmp < a[ j - gap ]; j -= gap )
                a[ j ] = a[ j - gap ];
            a[ j ] = tmp;
        }
}
```

heapSort

```
@Assertion(value = { @Case(
pre = "a !=null",
post = "isSorted(a)") })
public static void heapsort( Comparable [ ] a )
{
    for( int i = a.length / 2; i >= 0; i-- )
```

```

        percDown( a, i, a.length );
    for( int i = a.length - 1; i > 0; i-- )
    {
        swapReferences( a, 0, i );
        percDown( a, 0, i );
    }
}

```

mergeSort

```

@Assertion(value = { @Case(
pre = "a !=null",
post = "isSorted(a)" })
public static void mergeSort( Comparable [ ] a )
{
    Comparable [ ] tmpArray = new Comparable[ a.length ];
    mergeSort( a, tmpArray, 0, a.length - 1 );
}

```

quickSort

```

c.    @Assertion(value = { @Case(
pre = "a !=null",
post = "isSorted(a)" })
public static void quicksort( Comparable [ ] a )
{
    quicksort( a, 0, a.length - 1 );
}

```