Activity 1)

The method reverses the string s, for the length of (s).

```
String s = "Hello";
String t = "";
int x = 0;
/**
 * @updates x, t
 * @maintains t = rev(s)[0,x)
 * @decreases |s|- x
 **/
while (x < s.length())

{

   t = s.charAt(x) + t;

   x++;
}

out.println(t);
```

Activity 2)

```
int count = 0;
int idx = 0;
/**
 * @updates count, idx
 * @maintains s, ch, idx and count <= |s|

CORRECT: @maintains count = [number of ch values in substring s[0,idx)]


 * @decreases |s|- idx (the difference in size between s and idx)
 **/
while (idx < s.length()) {
    if (s.chartAt(idx) == ch) {
        count++;
    }
    idx++;
}
```

Activity 3)

```
int idx = start;
/**
 * (assume SEP is a global Set that holds whitespace chars)
 * @updates idx
 * @maintains s , the char at idx is the same as char at start

CORRECT: @maintains [s(start,idx} only contains whitespace OR contains no whi
-tespace char]


 * @decreases |s| - idx
 **/
while (idx < s.length() && SEP.contains(s.charAt(idx)) ==
                    SEP.contains(s.charAt(start))) {
        idx++;
}

//return s.substring(start,idx);
```

Activity 4)

```
       int lowEnough = 0;
       int tooHigh = n + 1;
        /**
         * @updates lowEnough, tooHigh
         * @maintains lowEnough is less than guess, which is less than tooHigh


correct: @maintains [n^1/r is within the bounds of [lowEnough,tooHigh)]


         * @decreases tooHigh - lowEnough
         */
        while (tooHigh - lowEnough > 1) {
            int guess = (tooHigh + lowEnough) / 2;
            if (n < power(guess, r)) {
                tooHigh = guess;
            } else {
                lowEnough = guess;
            }
        }
        return lowEnough;
```