Exercise 2

```
1   /**
2   * searches a given array for the serach value searchme
3   * @param array, a sorted array with values
4   * @param searchMe the search value
5   * @return If the value is in the array, then the return
        value, i,
6   *   is the index of the element, -1 otherwise.
7   */
8   public static int mySearch (int[] array, int searchMe) {
9
10      int minPos = 0;
11      int maxPos = array.length - 1;
12
13      if (maxPos >= 0) {
14          if ((array[minPos] > searchMe) || (array[maxPos] <
                searchMe)) {
15              // the search value is smaller or greater than
                    any element,
16              // so we know that the element cannot be in
                    there
17              return -1;
18          }
19      }
20
21      while (maxPos >= minPos) {
22          int middle = (minPos + maxPos) / 2;
23          if (array[middle] == searchMe) {
24              // searchMe has been found
25              return middle;
26          }
27          else if (array[middle] > searchMe) {
28              // eliminate positions >= middle
29              // search only the lower half
30              maxPos = middle - 1;
31          }
32          else {
33              // eliminate locations <= middle
34              minPos = middle + 1;
35          }
36      }
37
38      // maxPox < minPos, so searchMe cannot be found in the
            array
39      return -1;
40  }
```

V(G) = P+1 = 5+1 = 6

E - edges

N → nodes

V(G) = E - N + 2 = 17 - 13 + 2 = 6

Good cyclomatic complexity should be below 10, which this program satisfies. 10-20 is moderate risk, 20-50 is high risk and 50+ is really hight risk and considered as untestable.

CC shows us minimal number of needed tests for our method.