Let-Mechanism & Macros

```
3  public class HighScoreTable {
4      int[] scores = new int[10];
5
6      /**
7       * adds a new score to the high score list
8       * @param newScore newScore to add
9       * @return 0 when not added (since below 10 best)
10      *         2 if new high score
11      *         1 otherwise
12      */
13     public int addScore(int newScore) {
14         int min = 0;
15         boolean maxScore = true;
16         for (int i=0; i < 10; i++) {
17             if (scores[i] < scores[min]) {
18                 min = i;
19             }
20             if (newScore <= scores[i]) {
21                 maxScore = false;
22             }
23         }
24         if (newScore <= scores[min]) {
25             return 0;
26         }
27         scores[min] = newScore;
28         return maxScore ?2 :1;
29     }
30 }
```

Let-Mechanism & Macros

```
3  public class HighScoreTable {
4      int[] scores = new int[10];
5
6      /**
7       * adds a new score to the high score list
8       * @param newScore newScore to add
9       * @return 0 when not added (since below 10 best)
10      *         2 if new high score
11      *         1 otherwise
12      */
13     public int addScore(int newScore) {
14         int min = 0;
15         boolean maxScore = true;
16         for (int i=0; i < 10; i++) {
17             if (scores[i] < scores[min]) {
18                 min = i;
19             }
20             if (newScore <= scores[i]) {
21                 maxScore = false;
22             }
23         }
24         if (newScore <= scores[min]) {
25             return 0;
26         }
27         scores[min] = newScore;
28         return maxScore ?2 :1;
29     }
30 }
```

```
//@ requires newScore >= 0;
//@ ensures
      (\forall int i; 0<=i && i<10; newScore <= \old(scores[i]))
  ==> (\result == 0);
//@ ensures
      (\exists int i; 0<=i && i<10; newScore > \old(scores[i]))
   && (\exists int i; 0<=i && i<10; newScore <= \old(scores[i]))
  ==> (\result == 1);
//@ ensures
      (\forall int i; 0<=i && i<10; newScore > \old(scores[i]))
  ==> (\result == 2);
```

Let-Mechanism & Macros

```
3  public class HighScoreTable {
4      int[] scores = new int[10];
5
6      /**
7       * adds a new score to the high score list
8       * @param newScore newScore to add
9       * @return 0 when not added (since below 10 best)
10      *         2 if new high score
11      *         1 otherwise
12      */
13     public int addScore(int newScore) {
14         int min = 0;
15         boolean maxScore = true;
16         for (int i=0; i < 10; i++) {
17             if (scores[i] < scores[min]) {
18                 min = i;
19             }
20             if (newScore <= scores[i]) {
21                 maxScore = false;
22             }
23         }
24         if (newScore <= scores[min]) {
25             return 0;
26         }
27         scores[min] = newScore;
28         return maxScore ?2 :1;
29     }
30 }
```

```
minimalScore = (\some int score;
    (\exists int i; 0<=i && i<10; score == scores[i])
  &&(\forall int i; 0<=i && i<10; score <= scores[i]));

maximalScore = (\some int score;
    (\exists int i; 0<=i && i<10; score == scores[i])
  &&(\forall int i; 0<=i && i<10; score >= scores[i]));
```

```
//@ ensures newScore <= \old(minScore)
  ==> (\result == 0);
//@ ensures newScore > \old(minScore) && newScore <= \old(maxScore)
  ==> (\result == 1);
//@ ensures newScore > maxScore
  ==> (\result == 2);
```

Let-Mechanism & Macros

```
lowerThanMin(int score) = (\forall int i; 0<=i && i<10; score <= \old(scores[i]));
```

```
3  public class HighScoreTable {
4      int[] scores = new int[10];
5
6      /**
7       * adds a new score to the high score list
8       * @param newScore newScore to add
9       * @return 0 when not added (since below 10 best)
10      *         2 if new high score
11      *         1 otherwise
12      */
13     public int addScore(int newScore) {
14         int min = 0;
15         boolean maxScore = true;
16         for (int i=0; i < 10; i++) {
17             if (scores[i] < scores[min]) {
18                 min = i;
19             }
20             if (newScore <= scores[i]) {
21                 maxScore = false;
22             }
23         }
24         if (newScore <= scores[min]) {
25             return 0;
26         }
27         scores[min] = newScore;
28         return maxScore ?2 :1;
29     }
30 }
```

```
//@ ensures
      lowerThanMin(newScore)
  ==> (\result == 0);
//@ ensures
     !lowerThanMin(newScore)
   && lowerThanMax(newScore)
  ==> (\result == 1);
//@ ensures
      !lowerThanMax(newScore)
  ==> (\result == 2);
```