

## Instruction and Homework Assignment

### 7 Smurfless Smurfs

All smurfs of smurf country have visited the sauna. When they return to the dressing room, they find that the cat Azrael has mixed up their hats. Each smurf has a unique size of head and an exactly fitting hat. Unfortunately, all hats look, and smell, the same, so the only thing one can do is to try and fit a hat on a smurf and conclude one of three things: that it is too small, too big, or smurfs exactly.

Only Papa Smurf has found his hat immediately, for reasons beyond the scope of this exercise. So he takes charge of disentangling the mess. One option is to try every smurf-hat combination. This would take thousands of tries, however, since there are about one hundred smurfs. And imagine that Azrael returns in the meantime! Help Papa Smurf with an algorithm to give every smurf his own hat as quickly as possible.

You are going to approach this problem in several steps.

Given is the class `MadHatter` that simulates the fitting of the hats. The command

```
MadHatter mh = new MadHatter(N);
```

creates a dressing room mess with  $N$  smurfs and  $N$  hats. Smurfs and hats are both numbered from 0 to  $N - 1$ . The following methods are available to you.

- `int fit(int s, int h)` fits hat  $h$  on smurf  $s$ ; it returns 0 if the hat fits, 1 if it is too small and -1 if it is too large;
- `int compareSmurfs(int s1, int s2)` compares the size of the heads of smurf  $s1$  and  $s2$ . Returns 1 if  $s1$  has the biggest head, -1 if  $s2$  has the biggest head, and 0 if both are equally thickheaded (which would imply that  $s1$  and  $s2$  are equal);
- `int compareHats(int h1, int h2)` similarly for the size of two hats.
- `void matchtest(int[] hats)` Checks a solution and prints the results. The parameter `hats` should have length  $N$  and `hats[i]==j` expresses that smurf  $i$  should match hat  $j$ . If the solution is wrong, it prints a smurf-hat pair that doesn't match. Furthermore, it prints statistics about the usage of the compare and fit methods.
- `void printStatistics()` prints how many calls were issued to the compare and fit methods.

For example, suppose that smurf 3 has head size 52 and hat 0 has size 53. Then the call `mh.fit(3, 0)` will return -1.

All methods require that the smurf and hat numbers are in the domain  $[0 \dots N)$ . Errors will occur when you do not respect this.

1. Add the library file `MadHatter.zip` with `MadHatter.class` to your project as follows.

- Open your Netbeans project.
- Open the properties of your project by right-clicking on the project name (left pane).

- Choose “Libraries” and click the button “Add JAR/Folder”.
  - Select the Zip-file.
2. Suppose we can see which of two hats is the larger one and which of two smurfs has the bigger head. Write an efficient program that reads a number  $N$  of smurfs from input and then prints which smurf has which hat, in the format `smurf 3 fits hat 2` etc. Test your solution with the method `matchtest`.
  3. It is not easy to compare the sizes of smurf heads only by looking. So suppose the method `compareSmurfs` is not available. Again, devise an efficient program that gives each smurf his hat.
  4. Finally, assume that it is also impossible to compare smurf’s hats. The only way to get information is to take a specific smurf and a specific hat and see if they match (using the method `fit`). Write a program that tries all combinations to find a matching hat for each smurf.
  5. One can do significantly better than trying all combinations. Write a program that does this. Hint: use the partition technique from quicksort.