How to read the feedback.txt file

The project submission requires two steps. First, you submit your project using the *submit* command on the server, and second, you check the correctness of your program using the *verify* command. The *verify* command will produce a file, e.g. *feedback.txt*, that shows whether your program was compiled and run successfully, and whether your program produces the correct output when we supply a sample test case to your program. This manual will help you understand the feedback that you get in your *feedback.txt* file.

1. After running the command: `verify 90041 projA > feedback.txt` on the server, you will get feedback from our auto-testing system. The feedback is saved in the file that you specify in the command, e.g. in the above command, the name of the file is *feedback.txt*. The feedback shows the output of *sdiff* command that compares your output with the expected output.

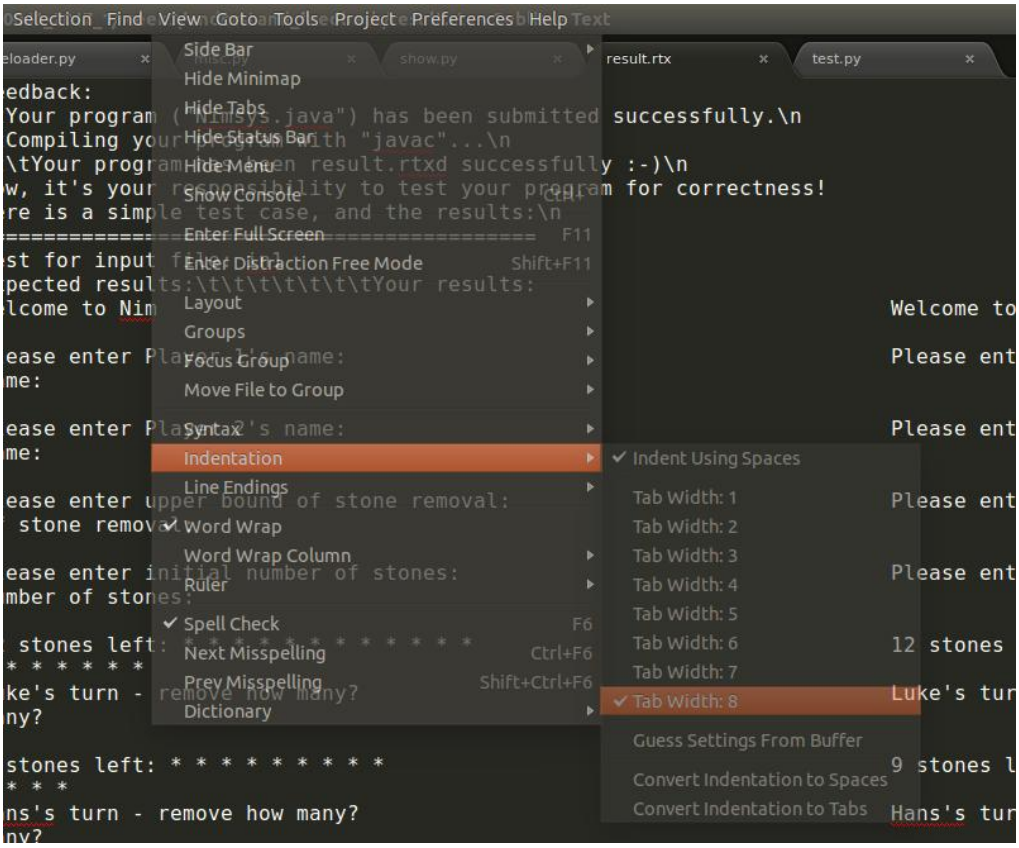
2. You may use *cat* or *more* commands to view the feedback file, that is by using one of the following commands:
   - `cat feedback.txt`
   - `more feedback.txt`

3. Alternatively, you may download the *feedback.txt* file to your computer and read it using a text editor. For example, you may use **Sublime Text** as a possible editor.

   With some of the text editors, the contents of the *feedback.txt* file could be misaligned, e.g. as shown below:

```
Feedback:
\tYour program ("Nimsys.java") has been submitted successfully.\n
\tCompiling your program with "javac"...\n
\t\tYour program has been result.rtxd successfully :-)\n
Now, it's your responsibility to test your program for correctness!
Here is a simple test case, and the results:\n
==========================================
Test for input file: in1
Expected results:\t\t\t\t\t\tYour results:
Welcome to Nim                              Welcome to Nim

Please enter Player 1's name:                       Please enter Player 1's name:

Please enter Player 2's name:                       Please enter Player 2's name:

Please enter upper bound of stone removal:              Please enter upper bound of stone remo

Please enter initial number of stones:                  Please enter initial number of stones:

12 stones left: * * * * * * * * * * * *             12 stones left: * * * * * * * * * * * *
Luke's turn - remove how many?                      Luke's turn - remove how many?

9 stones left: * * * * * * * * *                    9 stones left: * * * * * * * * *
Hans's turn - remove how many?                      Hans's turn - remove how many?
```

   The misalignment may be because of the indents generated by the *sdiff* command. The *sdiff* command uses TAB to indent, which assumes that editors show **TAB with width of 8**. As a result, to correct the feedback alignment, the TAB width of your favourite editor needs to be changed to 8. For example, if you are using Sublime Text, the screen-shot below shows how to do this:



4. If the autotest finds that your program's output matches the expected output ***exactly***, one of the lines in the *feedback.txt* will be: "**Your results seem to be CORRECT :-).**"

5. If your program's output does not match exactly with the expected output, you will get a message saying that your output is incorrect. In this case, the *feedback.txt* file will show you the differences as well.

   Several examples of the *sdiff* command are given below to help you interpret the differences shown in the feedback. For more details about ***sdiff***, visit this webpage.

   In case of incorrect output or other errors, make required changes to your program, and please submit and verify your project again. You can repeat this process until your program's output matches the expected output. You can submit as many times as you like **before** the deadline.

Example 1:
File a.txt has the following two lines:

```
apple
orange
```

and b.txt contains one line:

```
apple
```

sdiff *a.txt b.txt* will show the difference as:

```
apple          apple
orange   <
```

the < means that “the files differ and only the first file contains the line”.

Example 2:

File a.txt has two lines:

```
apple
orange
```

and c.txt's second line is different:

```
apple
banana
```

sdiff a.txt c.txt will show the difference as:

```
apple          apple
orange     |   banana
```

The | means that “the corresponding lines differ”.

Example 3
This one is trickier. File a.txt has two lines:

```
apple
orange
```

and d.txt has almost the same content, but without the third blank line, i.e., a.txt has a '\n' in the end of second line, and d.txt does not have '\n' in the end of second line:

```
apple
orange
```

the *sdiff a.txt d.txt* will show the difference as:

```
apple          apple
orange    /  orange
```

The / means that “the corresponding lines differ, and only the second line is incomplete”, where *incomplete line* means that a line does not end with a newline character '\n'.

We can use a hex/binary code viewer, e.g. by using *xxd* command on Engineering Server to view the binary representation of a.txt and d.txt. See the following:

xxd a.txt will give us:

```
6170 706c 650a 6f72 616e 6765 0a
```

xxd d.txt will give us:

```
6170 706c 650a 6f72 616e 6765
```

where *0a* is the ASCII hexadecimal representation of newline character '\n'. For more about ASCII code, read this webpage.