Reminder : Remember that for the duration of the defence, no segfault, nor other unexpected, premature, uncontrolled or unexpected termination of the program, else the final grade is 0. Use the appropriate flag. This rule is active thoughout the whole defence.

Author file

Check that the author file is at the root of the repository and formatted as explained in the subject. If not defence is finished and final grade is 0.

Yes | No

Memory leaks

Throughout the defence, pay attention to the amount of memory used by push_swap (using the command top for example) in order to detect any anomalies and ensure that allocated memory is properly freed. If there is one memory leak (or more), the final grade is 0.

Yes | No

Checker program - Error management

In this section, we'll evaluate the checker's error management. If at least one fails, no points will be awarded for this section. Move to the next one.

- Run checker with non numeric parameters. The program must display "Error".

- Run checker with a duplicate numeric parameter. The program must display "Error".

- Run checker with only numeric parameters including one greater than MAXINT. The program must display "Error".

- Run checker without any parameters. The program must not display anything and give the prompt back.

- Run checker with valid parameters, and write an action that doesn't exist during the instruction phase. The program must display "Error".

- Run checker with valid parameters, and write an action with one or several spaces before and/or after the action during the instruction phase. The program must display "Error".

Yes | No

Checker program - False tests

In this section, we'll evaluate the checker's ability to manage a list of instructions that doesn't sort the list. Execute the following 2 tests. If at least one fails, no points will be awarded for this section. Move to the next one.

Don't forget to press CTRL+D to stop reading during the intruction phase.

- Run checker with the following command "$>./checker 0 9 1 8 2 7 3 6 4 5" then write the following valid action list "[sa, pb, rrr]". Checker should display "KO".

- Run checker with a valid list as parameter of your choice then write a valid instruction list that doesn't order the integers. Checker should display "KO". You'll have to specifically check that the program wasn't developed to only answer correctly on the test included in this scale. You should repeat this test couple of times with several permutations before you validate it.

Yes | No

Checker program - Right tests

In this section, we'll evaluate the checker's ability to manage a liss of instructions that sort the list. Execute the following 2 tests. If at least one fails, no points will be awarded for this section. Move to the next one.

Don't forget to press CTRL+D to stop reading during the instruction phase.

- Run checker with the following command "$>./checker 0 1 2" then press CTRL+D without writing any instruction. The program should display "OK".

- Run checker with the following command "$>./checker 0 9 1 8 2" then write the following valid action list "[pb, ra, pb, ra, sa, ra, pa, pa]". The program should display "OK".

- Run checker with a valid list as parameter of your choice then write a valid instruction list that order the integers. Checker must display "OK". You'll have to specifically check that the program wasn't developed to only answer correctly on the test included in this scale. You should repeat this test couple of times with several permutations before you validate it.

Yes | No

Push_swap - Identity test

In this section, we'll evaluate push_swap's behavior when given a list, which has already been sorted. Execute the following 3 tests. If at least one fails, no points will be awarded for this section. Move to the next one.

- Run the following command "$>./push_swap 42". The program should display nothing (0 instruction).

- Run the following command "$>./push_swap 0 1 2 3". The program should display nothing (0 instruction).

- Run the following command "$>./push_swap 0 1 2 3 4 5 6 7 8 9". The program should display nothing (0 instruction).

Yes No

Push_swap - Simple version

If the following test fails, no points will be awarded for this section. Move to the next one.

- Run "$>ARG="2 1 0"; ./push_swap $ARG | ./checker $ARG". Check that the checker program displays "OK" and that the size of the list of instructions from push_swap is 2 OR 3. Otherwise the test fails.

Yes No

Another simple version

Execute the following 2 tests. If at least one fails, no points will be awarded for this section. Move to the next one.

- Run "$>ARG="1 5 2 4 3"; ./push_swap $ARG | ./checker $ARG". Check that the checker program displays "OK" and that the size of the list of instructions from push_swap isn't more than 12. Kudos if the size of the list of instructions is 8.

- Run "$>ARG="<5 random values>"; ./push_swap $ARG | ./checker $ARG" and replace the placeholder by 5 random valid values. Check that the checker program displays "OK" and that the size of the list of instructions from push_swap isn't more than 12. Otherwise this test fails. You'll have to specifically check that the program wasn't developed to only answer correctly on the test included in this scale. You should repeat this test couple of times with several permutations before you validate it.

Yes No

Push_swap - Middle version

If the following test fails, no points will be awarded for this section. Move to the next one.

- Run "$>ARG="<100 random values>"; ./push_swap $ARG | ./checker $ARG" and replace the placeholder by 100 random valid values. Check that the checker program displays "OK" and that the size of the list of instructions from push_swap isn't more than 700. Otherwise this test fails. You'll have to specifically check that the program wasn't developed to only answer correctly on the test included in this scale. You should repeat this test couple of times with several permutations before you validate it.

Yes No

Push_swap - Advanced version

If the following test fails, no points will be awarded for this section. Move to the next one.

- Run "$>ARG="<500 random values>"; ./push_swap $ARG | ./checker $ARG" and replace the placeholder by 500 random valid values (One is not called John/Jane Script for nothing) Check that the checker program displays "OK" and that the size of the list of instructions from push_swap isn't more than 5300. Otherwise this test fails. You'll have to specifically check that the program wasn't developed to only answer correctly on the test included in this scale. You should repeat this test couple of times with several permutations before you validate it.

Yes | No

Bonus

Reminder : Remember that for the duration of the defence, no segfault, nor other unexpected, premature, uncontrolled or unexpected termination of the program, else the final grade is 0. Use the appropriate flag. This rule is active throughout the whole defence. We will look at your bonuses if and only if your mandatory part is EXCELLENT. This means that you must complete the mandatory part, beginning to end, and your error management needs to be flawless, even in cases of twisted or bad usage. So if the mandatory part didn't score all the point during this defence bonuses will be totally IGNORED.

Some bonuses

To deserve being considered as such, a bonus must be:
- Useful (you will judge for yourself), no need to exaggerate.
- Well done
- Operational, it cannot generate any errors.

ex: -v to display the stacks during processing, -c to display with colours, etc...