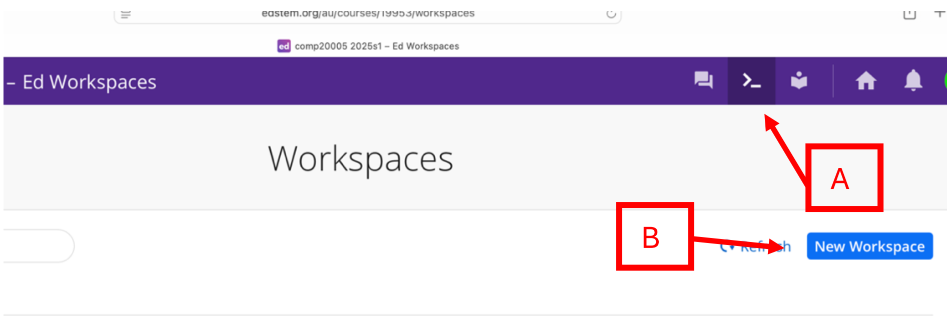
School of Computing and Information Systems
comp10002 and comp20005

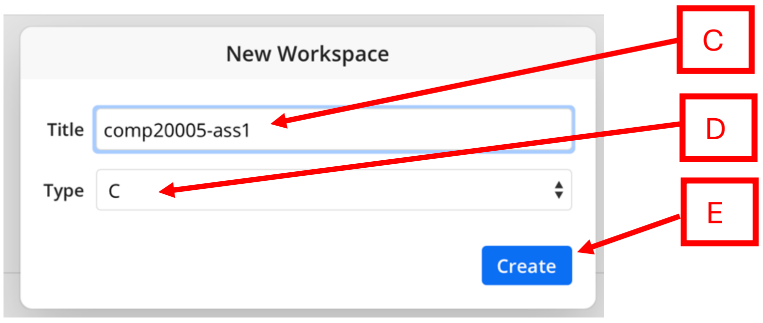
Using Ed Lessons Part B: Workspaces for Programming Assignments

These instructions are illustrated with examples from the comp20005 Assignment 1 for 2025. Your test input and output files may differ if you are looking at them while you prepare to start a different assignment or are enrolled in a different subject or different semester.

Note that assignment submission is via a tool called Gradescope. Even if you don't want to read anything else in this handout, make sure you read #14 and #15 on the last page!

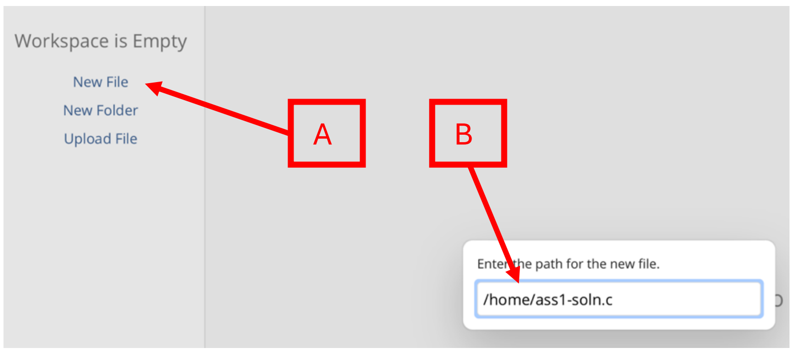
1: For each assignment create new workspace with a suitable name and file type…


edstem.org/au/courses/19953/workspaces
comp20005 2025s1 – Ed Workspaces
– Ed Workspaces
Workspaces
A
B
Refresh
New Workspace



Title: comp20005-ass1
Type: C

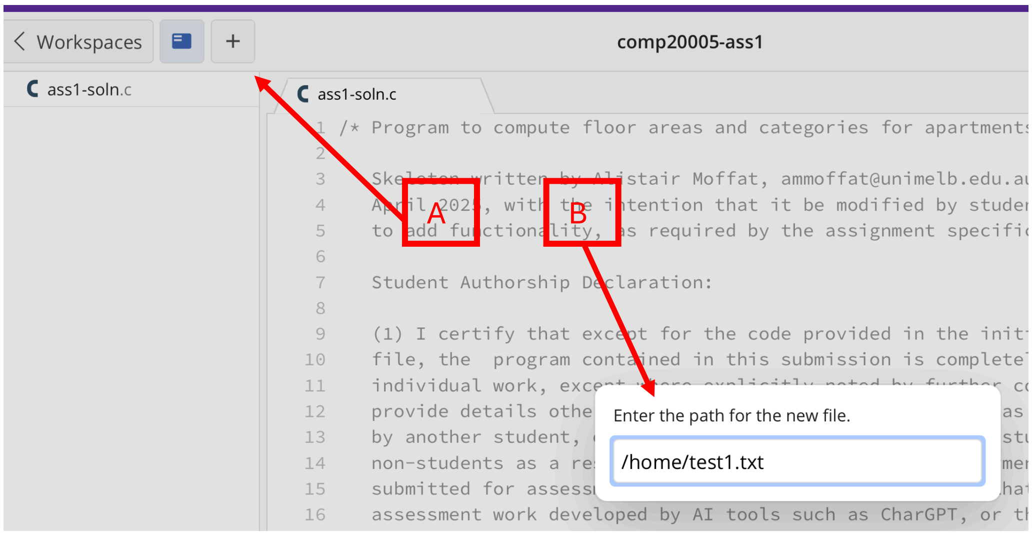
2: You'll get an empty workspace, with no files in it. Create a file for your C program…



Enter the path for the new file.
/home/ass1-soln.c

Access the assignment LMS page to copy the skeleton program, and paste it in as the starting point for your development…

3: Then use the same process to cut/paste each of the test data files into your workspace…

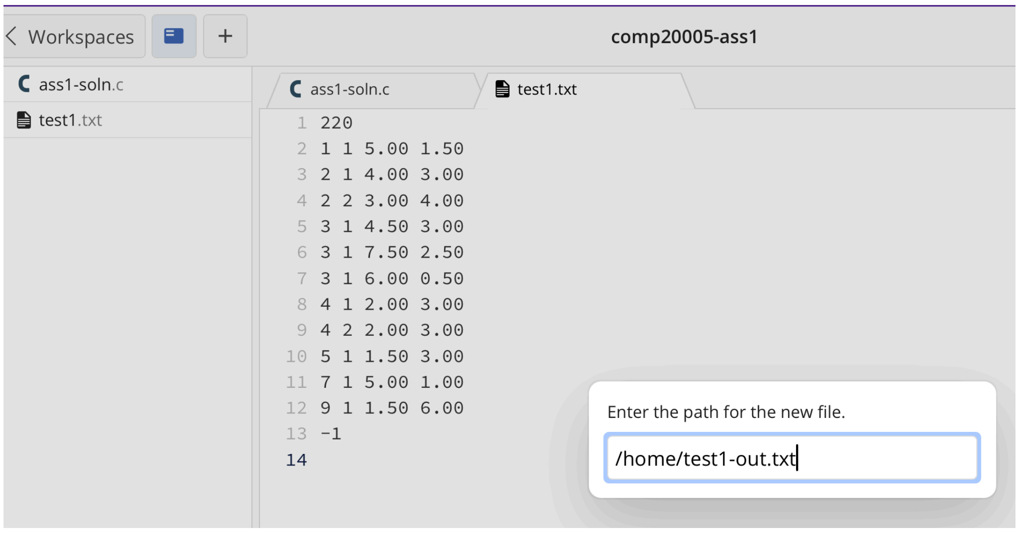


```
/* Program to compute floor areas and categories for apartments

   Skeleton written by Alistair Moffat, ammoffat@unimelb.edu.au
   April 2023, with the intention that it be modified by studer
   to add functionality, as required by the assignment specific

   Student Authorship Declaration:

   (1) I certify that except for the code provided in the init
   file, the  program contained in this submission is complete
   individual work, except where explicitly noted by further co
   provide details othe                                      as
   by another student,                                       stu
   non-students as a re                                      mer
   submitted for assess                                      hat
   assessment work developed by AI tools such as CharGPT, or th
```

Enter the path for the new file.

/home/test1.txt

4: And create the corresponding test output files too…

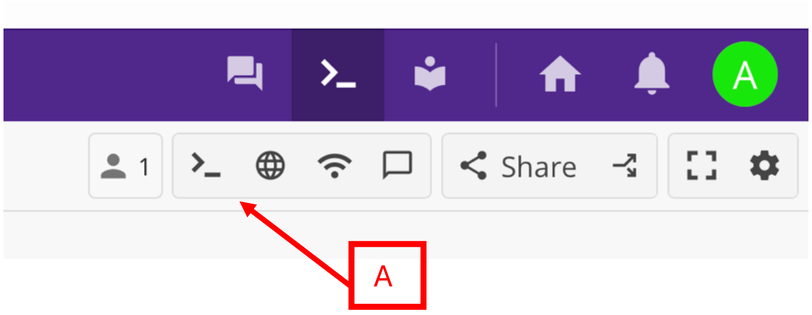


```
220
1 1 5.00 1.50
2 1 4.00 3.00
2 2 3.00 4.00
3 1 4.50 3.00
3 1 7.50 2.50
3 1 6.00 0.50
4 1 2.00 3.00
4 2 2.00 3.00
5 1 1.50 3.00
7 1 5.00 1.00
9 1 1.50 6.00
-1
```

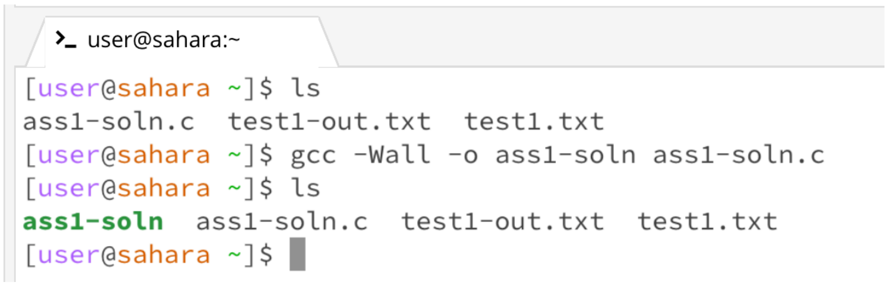
Enter the path for the new file.

/home/test1-out.txt

5: You can now start a shell…

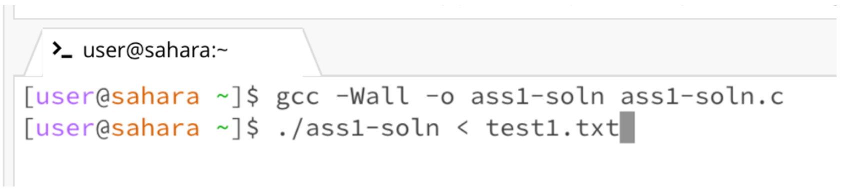


6: The shell accepts Unix commands, for example, use “ls” to list the files that are in the workspace, and “gcc” to check that you can compile the skeleton program…



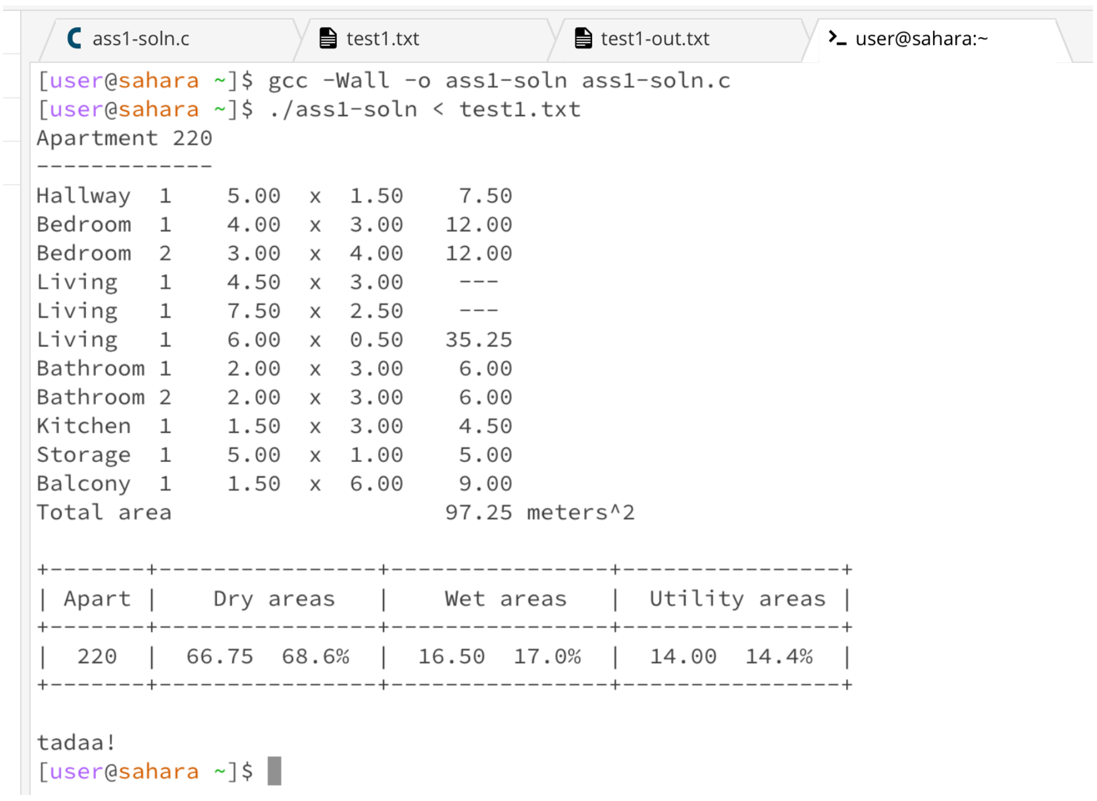
```
[user@sahara ~]$ ls
ass1-soln.c  test1-out.txt  test1.txt
[user@sahara ~]$ gcc -Wall -o ass1-soln ass1-soln.c
[user@sahara ~]$ ls
ass1-soln  ass1-soln.c  test1-out.txt  test1.txt
[user@sahara ~]$
```

7: Then, once you have written the first part of your code and are ready to start testing your program, you can execute it by taking input from one of the test.txt files…



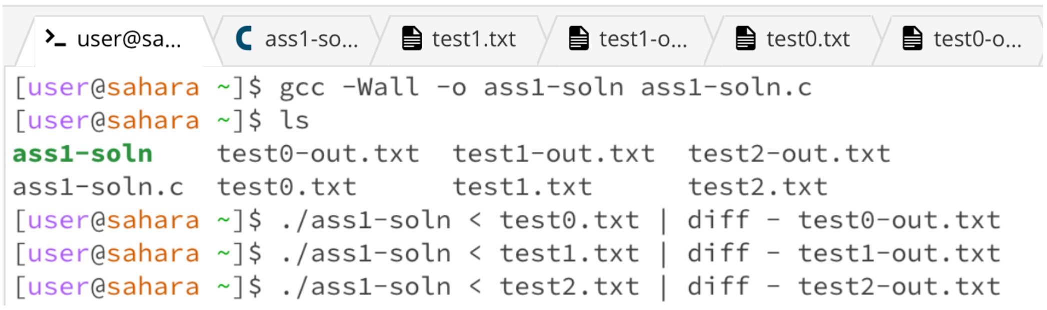
```
[user@sahara ~]$ gcc -Wall -o ass1-soln ass1-soln.c
[user@sahara ~]$ ./ass1-soln < test1.txt
```

8: While you are in the early stages of developing your program will probably just be looking at the output in the shell window like in this example (note that you can drag the shell tab to the top menu, to get more working room, like is shown here)…



```
[user@sahara ~]$ gcc -Wall -o ass1-soln ass1-soln.c
[user@sahara ~]$ ./ass1-soln < test1.txt
Apartment 220
-------------
Hallway  1    5.00  x  1.50    7.50
Bedroom  1    4.00  x  3.00   12.00
Bedroom  2    3.00  x  4.00   12.00
Living   1    4.50  x  3.00    ---
Living   1    7.50  x  2.50    ---
Living   1    6.00  x  0.50   35.25
Bathroom 1    2.00  x  3.00    6.00
Bathroom 2    2.00  x  3.00    6.00
Kitchen  1    1.50  x  3.00    4.50
Storage  1    5.00  x  1.00    5.00
Balcony  1    1.50  x  6.00    9.00
Total area                    97.25 meters^2

+-------+----------------+----------------+----------------+
| Apart |   Dry areas    |   Wet areas    | Utility areas  |
+-------+----------------+----------------+----------------+
|  220  |  66.75  68.6%  |  16.50  17.0%  |  14.00  14.4%  |
+-------+----------------+----------------+----------------+

tadaa!
[user@sahara ~]$
```

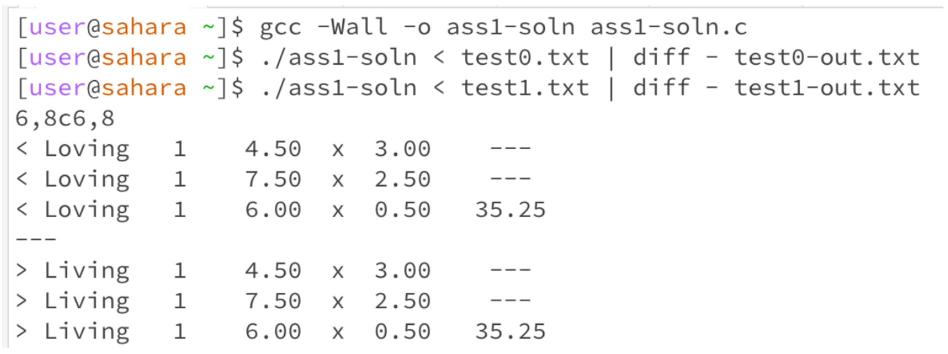
9: When you think you are done, it's time for the moment of truth – carry out a detailed check of your output against what *should* be generated for each of the test input/output pairings, using another Unix command called "diff"…



```
[user@sahara ~]$ gcc -Wall -o ass1-soln ass1-soln.c
[user@sahara ~]$ ls
ass1-soln     test0-out.txt  test1-out.txt  test2-out.txt
ass1-soln.c   test0.txt      test1.txt      test2.txt
[user@sahara ~]$ ./ass1-soln < test0.txt | diff - test0-out.txt
[user@sahara ~]$ ./ass1-soln < test1.txt | diff - test1-out.txt
[user@sahara ~]$ ./ass1-soln < test2.txt | diff - test2-out.txt
```

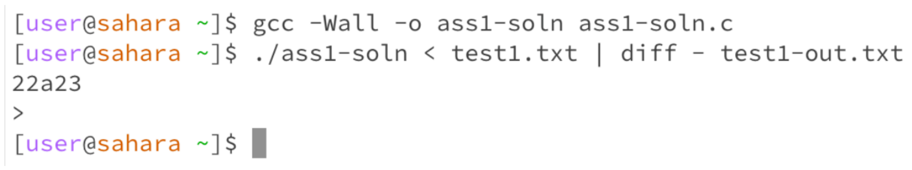
If "diff" is silent across the set of test files like is shown in this screenshot then your program generates exactly the required output, and you can shout *tadaa*! (But maybe you should also invent your own additional test data and try that too.)

10: On the other hand, if "diff" shows differences, well, you need to try and fix your program. To get this next screenshot a deliberate mistake was put into the program, with the program output (the "<" lines) now differing from the required output (the ">" lines)…



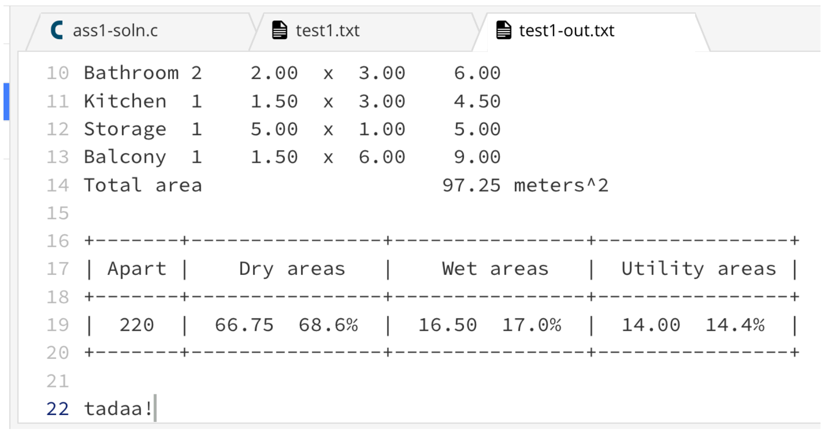
```
[user@sahara ~]$ gcc -Wall -o ass1-soln ass1-soln.c
[user@sahara ~]$ ./ass1-soln < test0.txt | diff - test0-out.txt
[user@sahara ~]$ ./ass1-soln < test1.txt | diff - test1-out.txt
6,8c6,8
< Loving   1    4.50  x  3.00    ---
< Loving   1    7.50  x  2.50    ---
< Loving   1    6.00  x  0.50   35.25
---
> Living   1    4.50  x  3.00    ---
> Living   1    7.50  x  2.50    ---
> Living   1    6.00  x  0.50   35.25
```

11: One problem that might be caused the file cut/pasting process is an extra newline added after the end of the sample output, which will show up in "diff" like this…



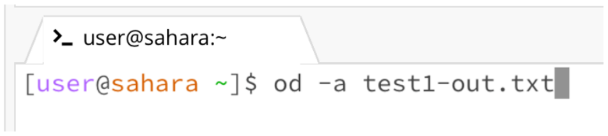
```
[user@sahara ~]$ gcc -Wall -o ass1-soln ass1-soln.c
[user@sahara ~]$ ./ass1-soln < test1.txt | diff - test1-out.txt
22a23
>
[user@sahara ~]$ █
```

To fix this problem, make sure that your copies of the test-out.txt files show the last line of required output as the last line of the file. There should be a single newline after the "tadaa!" message, which will look like this next screenshot in the Ed Lessons editing pane…



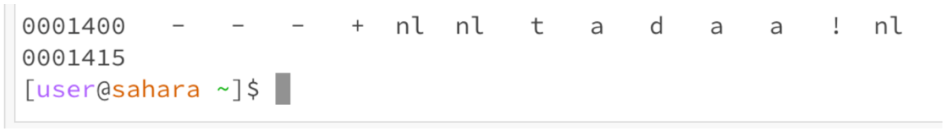
```
Bathroom 2    2.00  x  3.00     6.00
Kitchen  1    1.50  x  3.00     4.50
Storage  1    5.00  x  1.00     5.00
Balcony  1    1.50  x  6.00     9.00
Total area                     97.25 meters^2

+-------+-----------------+-----------------+----------------+
| Apart |    Dry areas    |    Wet areas    | Utility areas |
+-------+-----------------+-----------------+----------------+
|  220  |  66.75   68.6%  |  16.50   17.0%  |  14.00   14.4%  |
+-------+-----------------+-----------------+----------------+

tadaa!
```

12: Another Unix command that might help you get this right is “od -a”, which generates a byte-by-byte ASCII listing of a file…



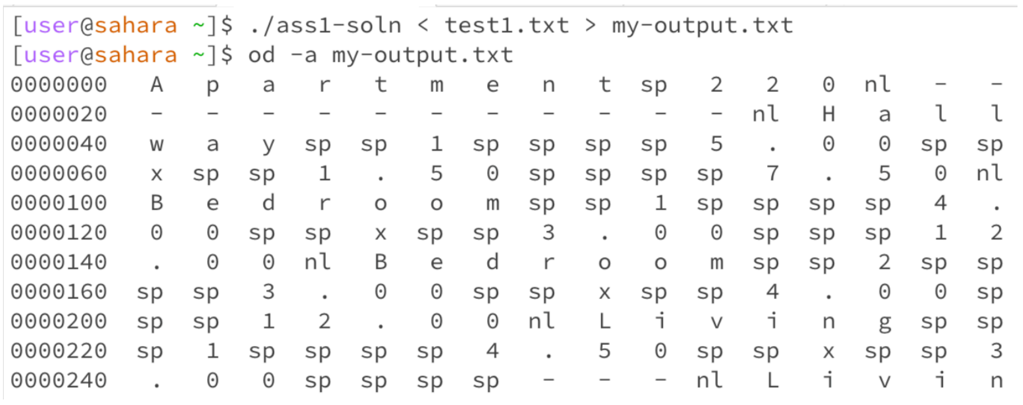
```
[user@sahara ~]$ od -a test1-out.txt
```

This is what you should see at the end of the “od” output if your test.txt files are correct, with a single newline (nl) after the final “t a d a a !”…



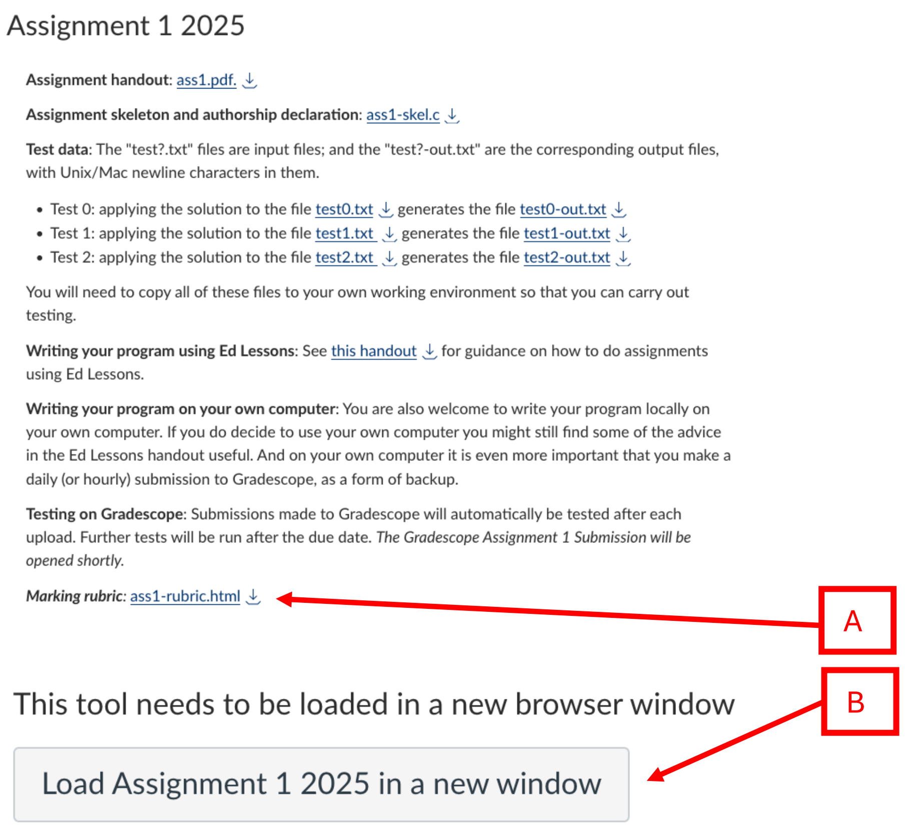
```
0001400   -   -   -   +  nl  nl   t   a   d   a   a   !  nl
0001415
[user@sahara ~]$
```

13: You can also save program output to a file and inspect it using “od” if you need to…



```
[user@sahara ~]$ ./ass1-soln < test1.txt > my-output.txt
[user@sahara ~]$ od -a my-output.txt
0000000   A   p   a   r   t   m   e   n   t  sp   2   2   0  nl   -   -
0000020   -   -   -   -   -   -   -   -   -   -   -  nl   H   a   l   l
0000040   w   a   y  sp  sp   1  sp  sp  sp  sp   5   .   0   0  sp  sp
0000060   x  sp  sp   1   .   5   0  sp  sp  sp  sp   7   .   5   0  nl
0000100   B   e   d   r   o   o   m  sp  sp   1  sp  sp  sp  sp   4   .
0000120   0   0  sp  sp   x  sp  sp   3   .   0   0  sp  sp  sp   1   2
0000140   .   0   0  nl   B   e   d   r   o   o   m  sp  sp   2  sp  sp
0000160  sp  sp   3   .   0   0  sp  sp   x  sp  sp   4   .   0   0  sp
0000200  sp  sp   1   2   .   0   0  nl   L   i   v   i   n   g  sp  sp
0000220  sp   1  sp  sp  sp  sp   4   .   5   0  sp  sp   x  sp  sp   3
0000240   .   0   0  sp  sp  sp  sp   -   -   -  nl   L   i   v   i   n
```

14: To submit your program (which is something you should do once a day through the assignment period, so that your development of your program is recorded, and also as a form of backup) you need to download a copy of your ass1-soln.c file to your computer (right click on the file in the Ed Workspace file menu and select “Download”; *be sure to get the “.c” file and not the compiled version*), and then submit it to Gradescope using the link at the bottom of the LMS Assignment page…


Assignment 1 2025

Assignment handout: ass1.pdf.

Assignment skeleton and authorship declaration: ass1-skel.c

Test data: The "test?.txt" files are input files; and the "test?-out.txt" are the corresponding output files, with Unix/Mac newline characters in them.

- Test 0: applying the solution to the file test0.txt generates the file test0-out.txt
- Test 1: applying the solution to the file test1.txt generates the file test1-out.txt
- Test 2: applying the solution to the file test2.txt generates the file test2-out.txt

You will need to copy all of these files to your own working environment so that you can carry out testing.

Writing your program using Ed Lessons: See this handout for guidance on how to do assignments using Ed Lessons.

Writing your program on your own computer: You are also welcome to write your program locally on your own computer. If you do decide to use your own computer you might still find some of the advice in the Ed Lessons handout useful. And on your own computer it is even more important that you make a daily (or hourly) submission to Gradescope, as a form of backup.

Testing on Gradescope: Submissions made to Gradescope will automatically be tested after each upload. Further tests will be run after the due date. *The Gradescope Assignment 1 Submission will be opened shortly.*

Marking rubric: ass1-rubric.html

This tool needs to be loaded in a new browser window

Be sure to double check that you have “signed” and dated the Authorship Declaration!

Then follow the usual Gradescope upload/submit process. After a few seconds you will be able to see what happened in Gradescope when we re-executed your program. *Be sure to check the Gradescope output, and if there are any issues then your program still has bugs*.

15: And maybe before you say “*done, I’ll get full marks for sure*”, read through the marking rubric linked from the LMS Assignment page, and ask yourself “have I done any of the things that will result in mark deductions?”

16: Tadaa!

Prepared by Alistair Moffat, ammoffat@unimelb.edu..au, April 2025