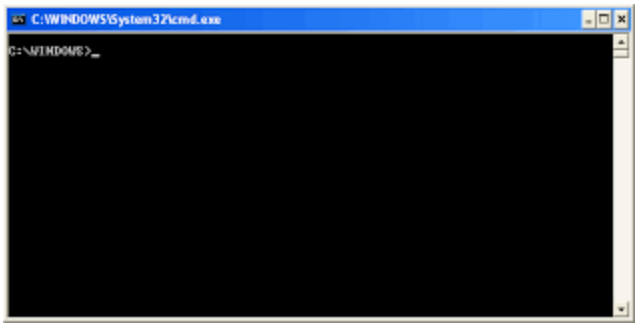
TESTING CS 1 PROGRAMS

Most of the input data for CS 1 programs will be quite large. It would be unreasonable to type in all of the data that we (CS1 staff) will use to test your programs. Thus, please follow these directions for testing each of your programs - both the individual programs and recitation/lab programs.

1) Use your IDE of choice and develop and compile your program there. A successful compilation should produce an executable (usually named prog.exe, prog, or a.out, where the source file was called prog.c). You can usually browse in the appropriate directory right after compilation and find the one newly created executable file.

2) Open up the Command Prompt in Windows or Terminal on a Mac. Here is a screen shot of the Command Prompt in Windows:



3) In the window, on the prompt, use the cd command to change directories to the location of our program. If you need to go "back" a directory, do "cd .." For example, if your program was in C:\COP3502 and the prompt was currently in C:\WINDOWS as shown above, you would enter the following at the command prompt:

```
C:\WINDOWS>cd ..
C:\>cd COP3502
C:\COP3502>
```

If the path is longer (nested in several directories) you can get to a directory in a single command instead of changing directories one folder at a time:

```
C:\>cd Users\Arup\Desktop
C:\Users\Arup\Desktop>
```

4) To run your program using input from a file, but where your program reads from the keyboard, keep the file in the same directory as the executable and type the following at the command prompt (in this example, we assume the program name is wordsearch and the input is stored in the file ws_sample.in, without a .txt at the end.):

```
C:\Users\Arup\Desktop>wordsearch.exe < ws-sample.in
```

5) What should happen is that all of the output should go to the screen in the Command Prompt window.

```
C:\Users\Arup\Desktop>wordsearch.exe < ws-sample.in
Words Found Grid #1:
trap
part
cats
Words Found Grid #2:
swing
wing
letter

C:\Users\Arup\Desktop>
```

6) This is fine and good if the output is short, but a pain to deal with if there's a ton of output. So, a better way to handle the output is to redirect it to a file. Instead of it appearing in the Command Prompt window all of the output will be written to a file with a name of your choice. Here we write all of the output to arup.out (from the previous example):

```
C:\Users\Arup\Desktop>wordsearch.exe < ws-sample.in > arup.out

C:\Users\Arup\Desktop>
```

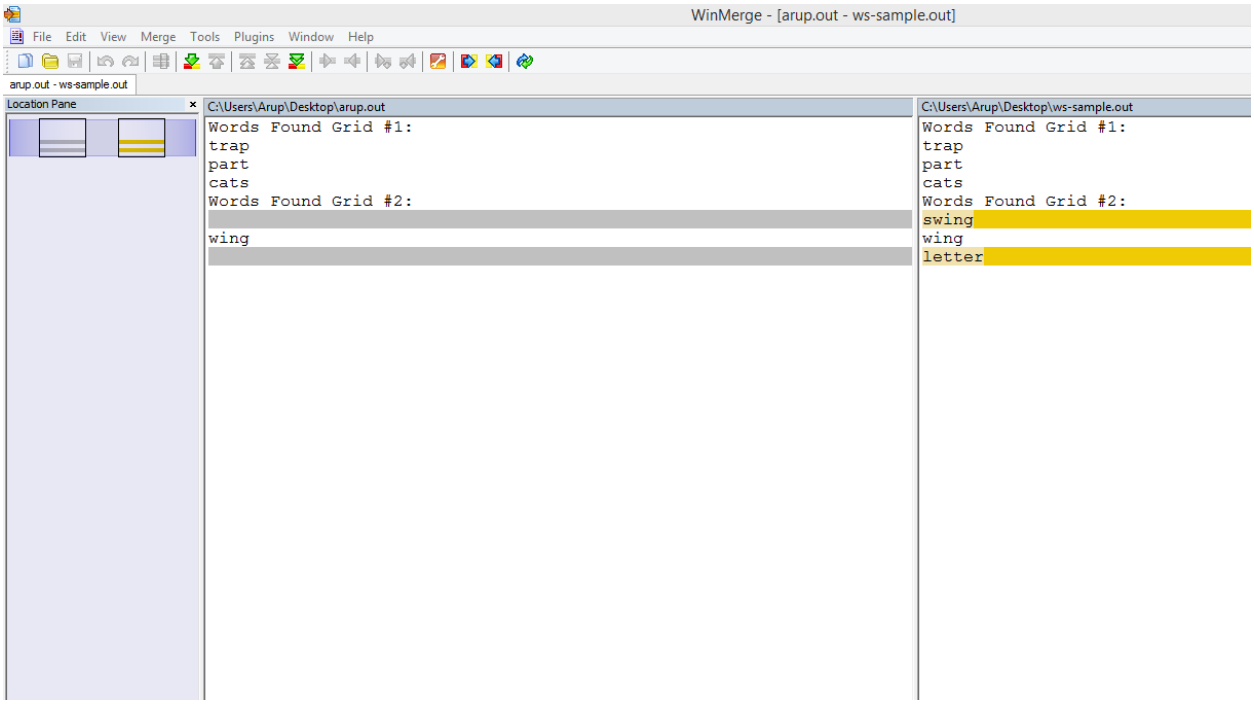
Notice how this time, no output shows up in the Command Prompt window. Instead, all of the output is in a new file called arup.out, which will be in the same directory as the executable program and input file.

7) Now that you have your program's output stored in a file, you can compare it with the correct output (assuming you have that stored in a file also). If your output is correct for the input you tested it on, here is what would happen:

```
C:\Users\Arup\Desktop>fc ws-sample.out arup.out
Comparing files ws-sample.out and ARUP.OUT
FC: no difference encountered
```

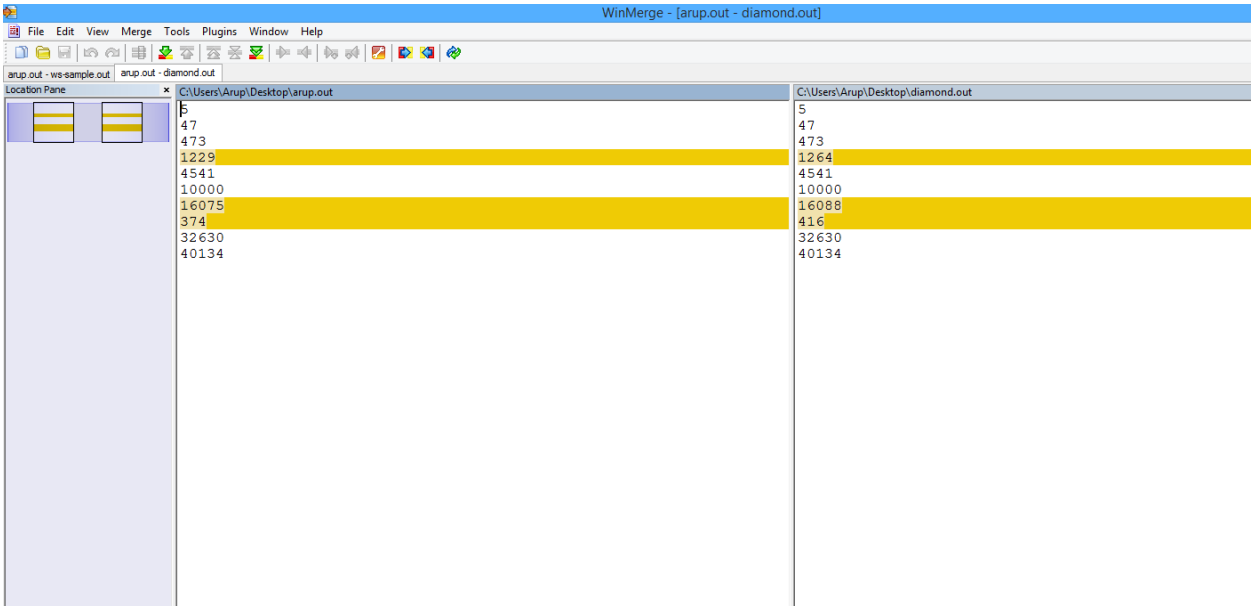
If this happens, great!

8) But…if this fc (the command for file compare) spits out some differences, you'll find that it's very difficult to read through these differences. In this case, you need to use a file comparison utility to compare the correct output file to the file that stores your program's output. WinMerge (http://winmerge.org/downloads/?lang=en) is one such free tool. After you download and install it, here is a screenshot of a comparison that has some errors in it:



In this case, what has happened is that my program did not find the words swing or letter in the second puzzle. The yellow highlights indicate which lines are incorrect.

Here is another case where a program output has the same number of lines as the correct output, but is incorrect:



This time you can see which cases are incorrect as the lines are highlighted in yellow. Using this information, you can proceed and debug your program, using the appropriate test cases. (For Mac users, at Terminal, unix commands must be used and a different file comparison utility.)