(20) Remember your assignment report!
1. Understanding: Show that you understand the weekly project and the overall lessons
1.1. Explain what the weekly project is asking you to do in English, pseudocode, and drawings
1.2. Explain, in your own words, what the overall lessons for the week were
2. Design: describe or draw out how the project components should behave
2.1. Use pseudocode, flowcharts, drawings, or explanations. This could be a combination of techniques
2.2. Describe the overall tasks and sub-tasks of your program
2.3. This design should at least consider any input, any processing, and any output that you think
you will need to solve the assignment requirements
3. Testing: Design and describe some tests to verify that your code would be working properly. Perhaps create
a table (input, expected output, and actual output) describing the tests that you plan to perform to
demonstrate that your program meets the assignment requirements.
4. Implementation: this will generally be the code you use to achieve your assignment requirements, but may
sometimes include provided files or other files you create.
5. Reflection: now that you are done with your program (even if the program is not complete!) you should
discuss the process. You should mention things like:
5.1. Was your understanding complete and design adequate at the start of the project or did you learn
something about the problem as you went. What details might someone have missed during the
understanding process that the implementation and testing process reveal?
5.2. Did all your tests work out the way you expected or did you have to alter your tests because of some
design, implementation, or testing details?
5.3. Did implementation go without any problems, were there details that were difficult to get working the
way you wanted them to?
5.4. What details might someone miss during the implementation process?
5.5. What techniques helped you approach the problem, from class and from outside class? Does this
project seem related to previous projects and do you see any names for future projects that it might be
related to?
Note: See the support files of the course for a simple example report.

Understanding-
This week for me was all about the basic syntax and semantics of C. Building these two programs have proven that without knowing syntax, programs can not run and without

semantics they run very, very difficultly. With both, programs run smoother and more appropriate. On Friday my group decided to write out all the pseudocode required for the gues_num program. None of it was good, I ended up scrapping it and creating my own. For the first program, the goal is straightforward, if room capacity is too much, then ask again for a different answer. For the two-player guessing game the goal is to have input, output and multiple checks and loops for the program to work effectively.

Design-
For this program, there needs to be an overarching while loop and two or more while loops inside to check for bad input. Once the input is decoded, it needs to be checked for correctness (player one's input). If it is correct, then the program needs to exit or loop back.

Testing-
For testing, I made multiple different inputs ensuring that the variables were being used correctly. This becomes a problem when large numbers are entered as well as strings. I also spent way too long trying to get the program to loop back and could not figure it out. I will have to resubmit a fully working one after I have gotten some help.

Reflection-
Since this was the first major assignment for this course, it was a bit of a learning curve. From thinking that it was going to be oh so easy on Friday with all the pseudocode already written out to getting home, opening my laptop and realizing the major problem- c does not have a string type. It also has no good way for input. This lead me down many google searches to find answers for these questions, and most of them turned out fine. The one problem that I could not solve was checking for equivalence. From checking if they ran out of tries and seeing if they wanted to continue playing, none of the equivalence statements like == seemed enough. Knowing Python certainly has been a double-edged sword, since some things are almost identical while others are wildly different. This has lead me to do stuff close to what I was doing last term, so much so that google came up with my search except for Python.