Hey everybody. I figured it would be a little easier for me to try and outline a few of my impressions regarding the program in document like this. I’ll put some trivial things at the end that deal mainly with formatting. If needed I can also help implement changes as needed. I just did not want to start shifting around a lot without getting a go ahead from the rest of the group.

So far using the base RPSGame that was posted on the Canvas site we seem to have a program that meets all the requirements. There are a few things that I have noticed, but I don’t know how far we want to go to try and repair them. I think that mainly this could serve as a something to consider later if any of this code has, or will be adapted into future projects.

# InputValidation

For the most part the input validation seems to work. I have noticed though, that it does not function for cases where the first character entered is the desired input. For instance, when choosing a tool to play if I input a garbage string like “sasfhj84484” the program still accepts this as valid input because it begins with an “s” that the program was looking for.

Another place I noticed this same kind of behavior with the InputValidation is while changing the strength for the tools. The program asks for integer input, but will accept decimals. I’m assuming that it is accepting the first portion of the input and discarding the rest.

Another thing I noticed was that the getType() method might have some issue with depreciated code. This applies to the scanf() method being used to assign the user input to the variable. I only noticed this when I was trying to compile this code in Visual Studio and it threw back some errors to use an updated method by swapping scanf() for scanf\_s(). For our purposes I don’t think this is a big deal, but I found a few articles that go over the differences in the two if you would like to read.

<http://stackoverflow.com/questions/21434735/difference-between-scanf-and-scanf-s>

<https://in.answers.yahoo.com/question/index?qid=20081030175426AAWMnHi>

It basically amounts to the newer version having ways built into it to prevent overwriting the array being used to store the string. With scanf it seems there is a possibility that if the user does input more than what is allocated for the buffer it will start writing to memory that does not belong to the array. This could lead to undefined behavior if this memory is then accessed later. Like I said earlier, not a huge deal for us now, but something to consider in the future as it might lead to a possible exploit of the program.

Unfortunately, I am not super familiar with this input validation method either so, I can’t really say the best way to fix it. I tried just changing the scanf to scanf\_s and that did not seem to work. The input validation I typically use functions a bit different from the stuff Harlan put up. I’m not as good at following some of his C stuff, so I’ve gone a different route.

# Small Stuff

* Following entering custom tool info for user there is no visual indication of the desired user input for the prompt about changing the computers tool strength
  + Prompt is shown below, no “y – yes, n – no” guidance
  + Should the computer use custom strengths for the tools?