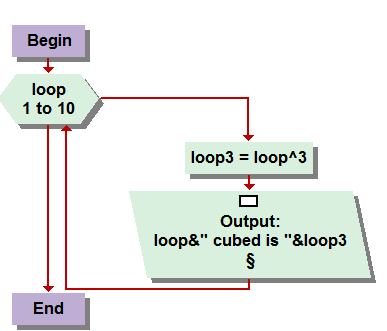
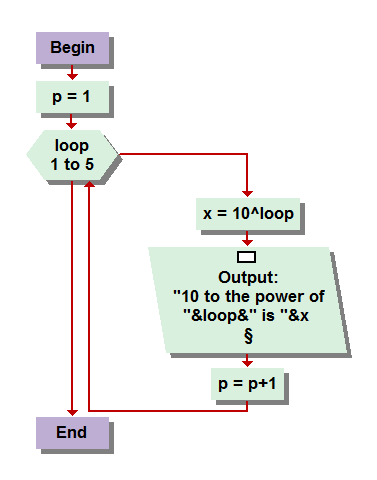


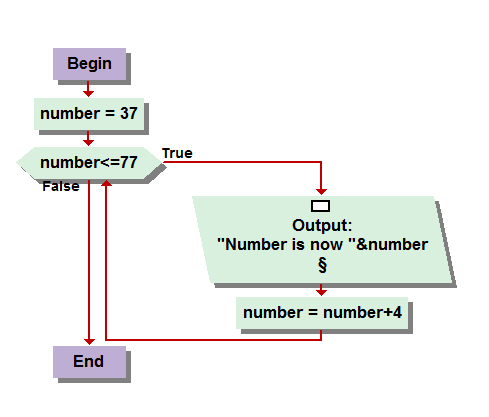
3.1   
I initially programed the solution to this problem using a for loop, but simply went back and added a counter into the algorithm enabling a while loop to be implemented. The count, or “ho,” starts at zero and each time a “ho” is outputted into the console, the loop adds one to the count, and only continues to count until the count reaches three. Then I directed the program to output Merry Christmas into the console, completing the assignment.



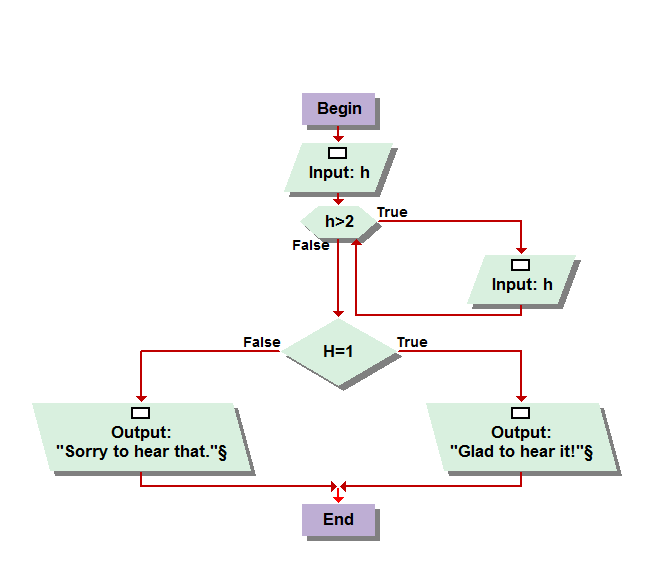
3.2  
This task was a tiny bit tricky. At first, I didn’t have the math quite right, just squaring the loop value in contrast to cubing it. Luckily I double checked it in the book and realized that I was in the wrong. I then made an assignment function to take the value of the loop at that time, raise it to the third power, and then output it in the console with the appropriate language to have it mirror the books desired solution.



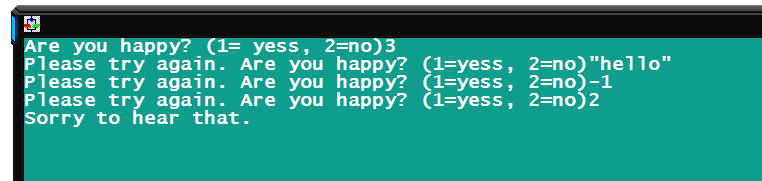
3.3   
Upon revisiting this, I see that I have an entirely useless variable of “P” in the equation that I started using for something but cant seem to remember what for. Regardless, I did encounter some programming set backs with this assignment, specifically with mention to my function for x. For one reason or another, I kept getting a scripted error from VL trying to, but not very well, explain to me that the way I had it written out was not Kosher. Although I believe all I did was close VL and re-write the program the same way and it worked the next time. As I mentioned on the boards I was having a little difficulty with my VL not quite following orders this week, lol. First with this and next with my console inputs, but after I closed everything and restarted my computer for, what was probably the first time in weeks, things seemed to be running smoother. It was pretty simple after the last one. Loop one through five, then raise ten to the value of the loop, and output into the console, the result of the function with the appropriate dialogue. Really enjoying these more sophisticated outputting’s.

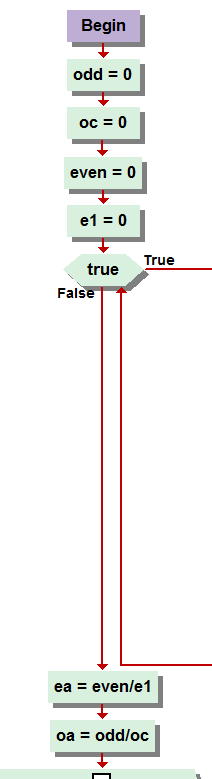
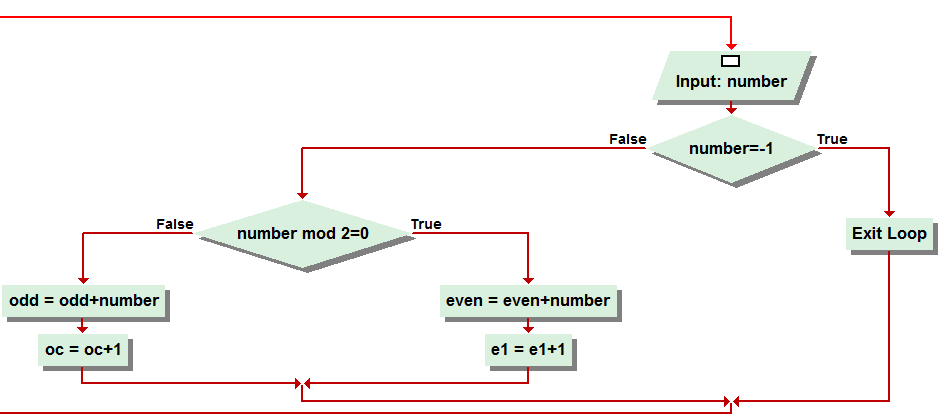
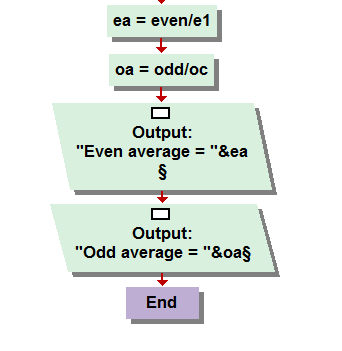


3.4  
Counting up with a while loop, I believe there was a demonstration of this in the book or a lecture, I do recall not having to work to hard on this one for the desired result. I programmed Number to begin at 37, and a while loop to out-put the number and add 4 to it , and provided the value of the new number was less than or equal to 77, it would repeat the process until such a result is reached. Very simplistic output statement, as asked by the book for this problem, so I didn’t consume too much of time.

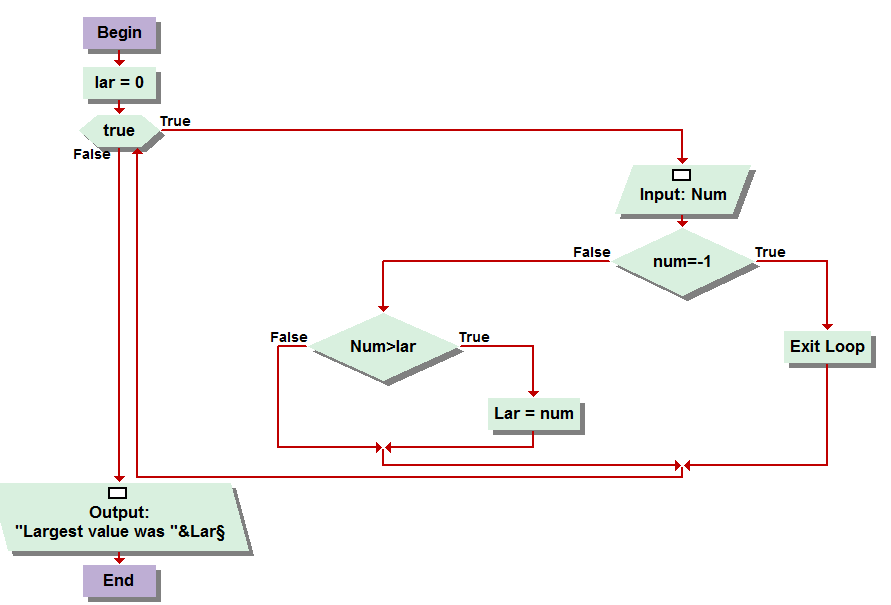


3.5  
After looking back at this, I now realize that my solution is only contingent to the input of positive integers. Any negative integer will skew the accuracy of the result, but is very simply corrected by adding the word “or” in between (H>2) “or” (h<0). Adding this stipulation to the condition would account for any negative integers while still giving the desired result . I just updated mine and tested it, and posted the new resulting console below for you to see.

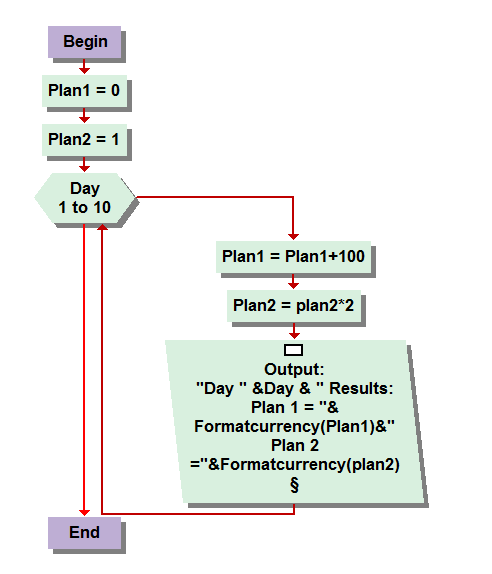




3.6  
The even odd average. This solution took a bit of fidgeting. This is the one I encountered the most difficulty with, mainly because of my console input not responding after opening and closing it two or three times to troubleshoot. I created a sentinel, of negative one, as instructed by the problem. I also created a long list of variables, as you can see… ODD, O(dd) C(ounter), even, and E(ven)1(counter), and the input of “number.” If the number = -1, the program would exit loop. Otherwise, it would be conditioned to “Mod 2=0” allowing the program to determine whether it is even or odd. If it is even, the value of variable “even” would now add the number to it. As well, add one value to the variable E1, which I had to change from EC, even counter, because of a nonsensical error that was presenting itself over and over saying it could not compute it for EC, though it could OC, and E1, but not EC. So I changed it. The same process would be conducted for odd numbers as well, and then processed to divided the sum of all, either even or odd, numbers entered by the total number of individual numbers entered, kept track by the appropriated counters. Then, the averages are outputted in the same console box as they were entered. After working out the minor kinks, this was relatively simple to develop.



3.7  
This problem was also very simplistic. You created a variable for Largest (lar) then a while loop terminated by a sentinel value. The value being negative one, if entered, otherwise, the number was conditioned against the variable “lar” ascertaining which one was in fact the largest. If the number entered was larger then the value representing “lar” than it would now become “lar” otherwise the input would prompt again until killed by the sentinel. At which point, a console output would display “the largest value entered was,”lar” revealing the largest of the values.



3.8  
Here was what appeared, at first glance, to be a very difficult problem to program. So I had a wild idea to just try something as simplistic as possible, and then if, which I imagined it wouldn’t, didn’t work, I would build on it until it did. However, the product of my own genius prevailed again, and in less than five minutes I had this one solved. I created two variables, plan1 and plan2 and a for loop for day 1 through ten. Again I could have used a while loop with a counting function, this was just more concise. Plan one started at zero and was looped to add 100 dollars each day, while plan 2 started at one dollar and was instructed to double each time the loop was ran, 1 through 10. After each of the mathematical aspects was completed for that sequence of loopage, they were both outputted in the appropriate order, reluctantly displaying the exact desired results as illustrated in the book. I thought it would take much more engineering but clearly I was over thinking and over worrying myself for no reason.