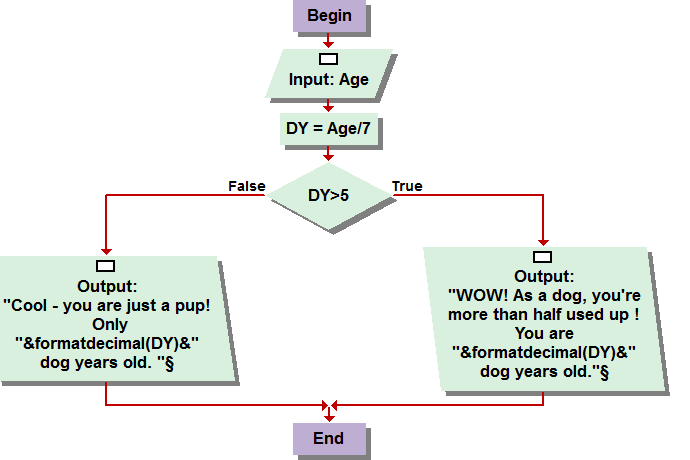
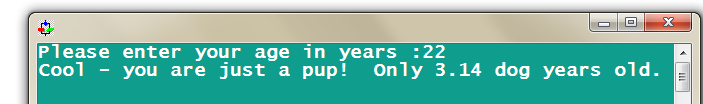
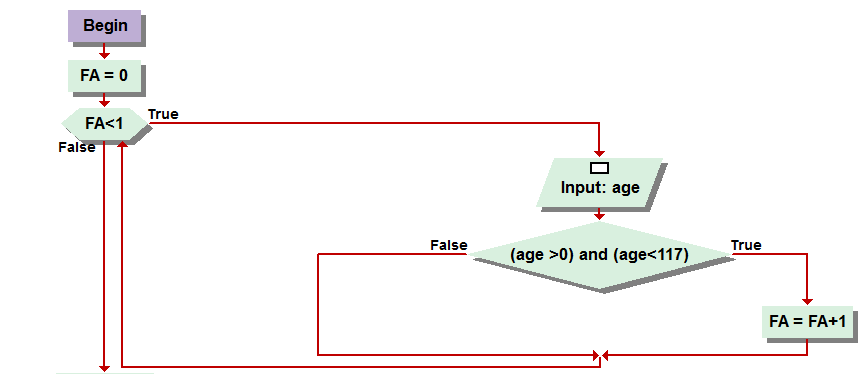
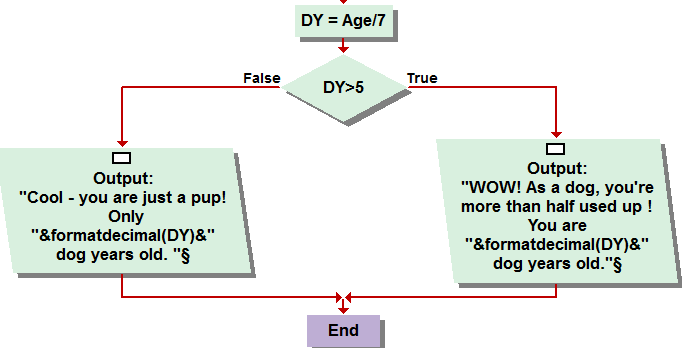
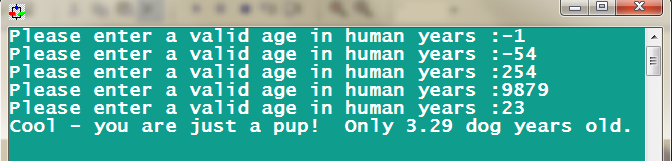
|  |  |  |  |
| --- | --- | --- | --- |
| Human years | Conversion | Dog years | Young vs. old |
| 7 | /7 | 1 | <5 |
| 21 | /7 | 3 | <5 |
| 46 | /7 | 6.57 | >5 |
| 225 | /7 | 32.14 | >5 |



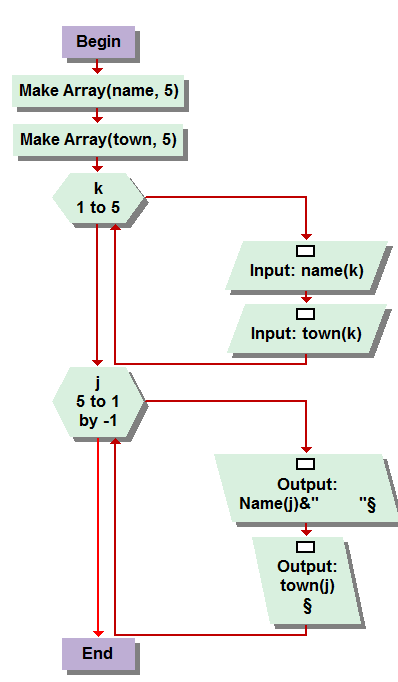
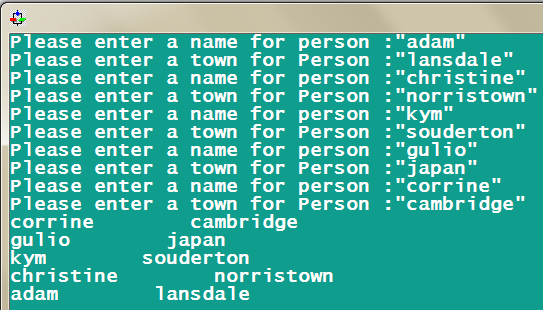
The new data, now converted into a dog years, then prompts the program to display one of two outputs. Contingent on whether the age in dog years is greater than or less than five, the output statement is determined. If the age is greater than five, a message informing the user that they are now, “more than half used up.” Additionally, the output statement also displays the converted age inputted into a decimal format in dog years. If the age converted is determined to be less than five, a message informing the user that, at that age, “Cool – you are just a pup!.” As well, the age is displayed in a decimal format in dog years.

Question 1:  
  
The first question was a simplistic conversion and output statement, contingent upon the input variable value of human years. Once the program user enters his age in human years, the program then divides the years by seven, equating the data into dog years.





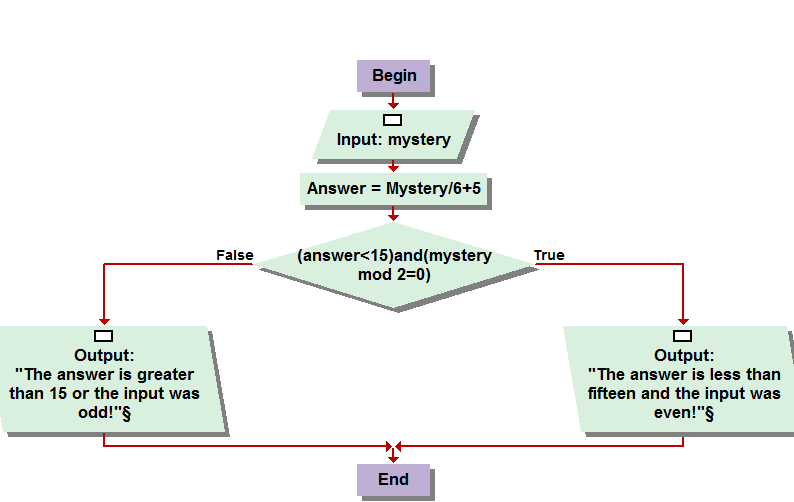
Question 2: The second problem was a remodel from the first, in the manner that it would continually prompt the user to enter a variable for the conversion, abiding that it is positive, and a reasonable age that a human could live until. Being that, I assume, the oldest person on record to live died at a hundred and seventeen years of age, that is the cut off for the conditional statement. A while loop is used to prompt for the input, conditionally looping while “FA,” or Found age equals zero. The variable FA is assigned to zero in the beginning of the program and will not be changed to anything else unless the input age meets both conditions of the “if” statement, henceforth, the value of the variable age is greater than zero and less than one hundred and seventeen. If this occurs, the sentinel value is triggered killing the loop, moving the program down to second half, which is the same from the first problem. Another If statement evaluates if the conversion renders the age in dog years less than or greater than five, displaying the same, appropriate output statement.

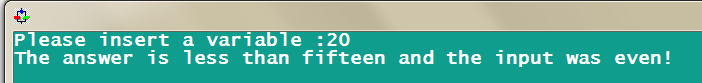


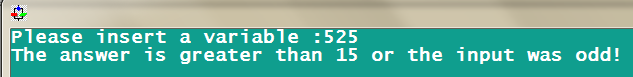
Question 3:  
This question tested the students ability to implement and sort arrays. The first element of the problem involved the creation of two parallel arrays. One for the name of the people inputted, and one for the town they are from. Both of these arrays have an upper bound of five, meaning that the for loops used to work within the arrays simply, would not need to exceed “upper bounds,” if you will, of five either, in this instance. The first for loop prompts for the two arrays, one names and one towns. They are sorted into their appropriate element by the value of which the loop is executing at the time of input. Once All the necessary inputs have been made into the arrays, henceforth rendering them complete, another for loop is implemented. This one, how-ever, will occur in reverse order from five to one. At the time each value of the loop is executed, starting from high to low, it will output the name and town of the corresponding array element. This, in actuality, simply causes the data to be “printed” in the reverse order of which it was entered. Sadly, I see now that I misspelled my sister’s name in the program. >.<

Question 4

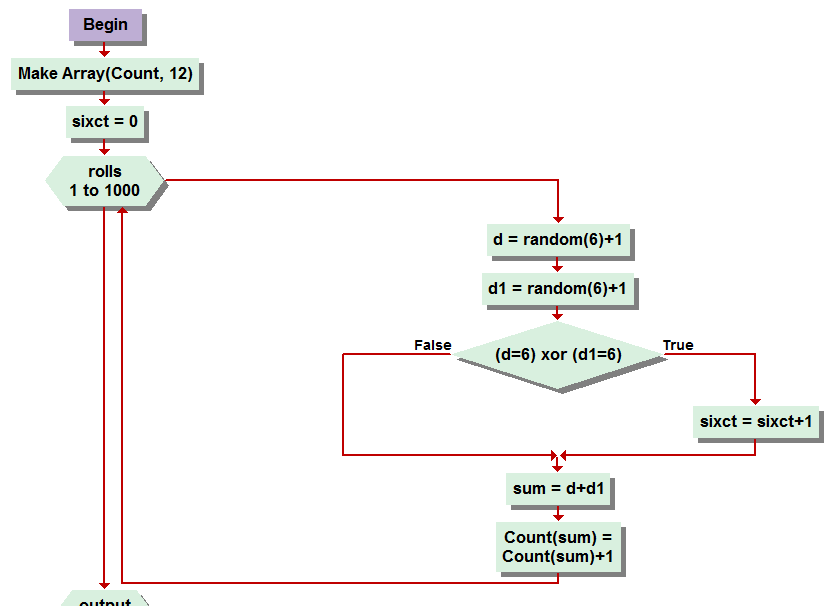
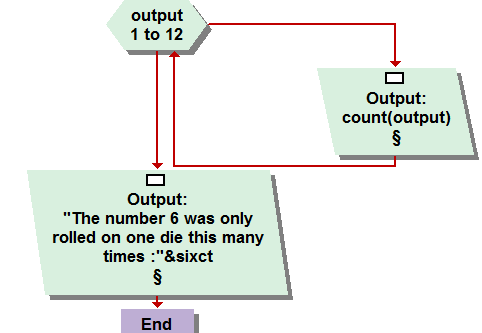
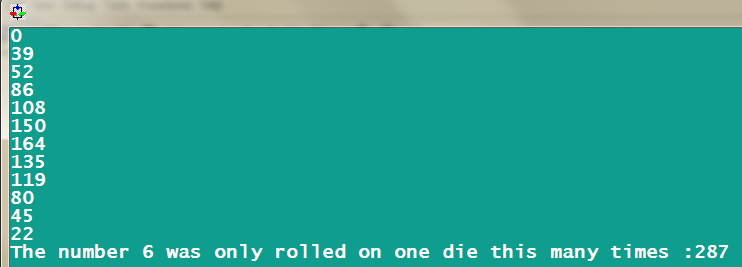
The three things that occur when an input statement are executed are essential to understanding the visual logic programming language on a whole. The first is the reservation of memory. This means that however much memory is needed from your computer’s RAM is preserved so that the variable can be referenced without the computer running too low on memory, leading to a crash and loss of data. The second thing to occur is the symbol recognition. The variable is associated with the input given, whether it be in dialog box, console, or file format, some type of information is entered to represent a variable and that is saved so it can be referenced throughout the life span of the program it exists within. The final procedure is initialization. Executing an input, although an input is not necessarily essential to every program’s existence, initializes other assignments, tests and loops to function if they are dependent on the existence of the input’s variable value.



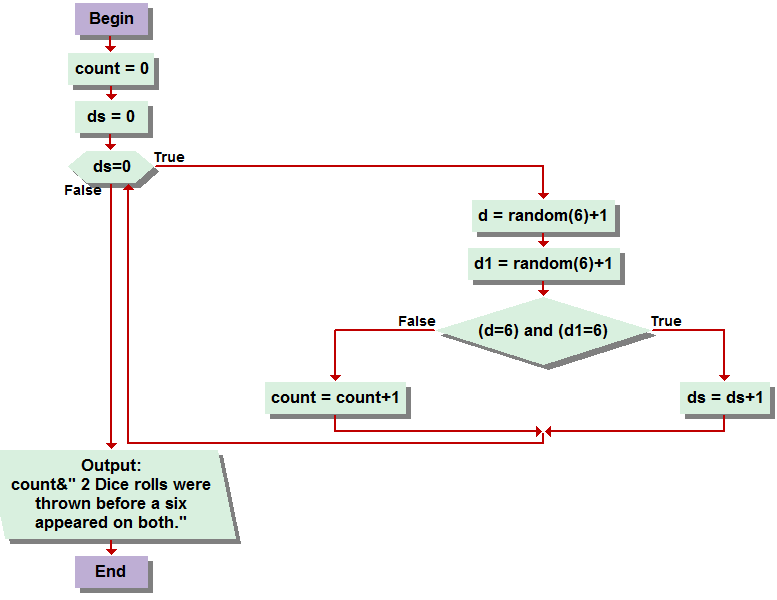
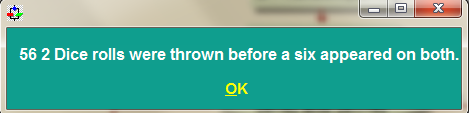
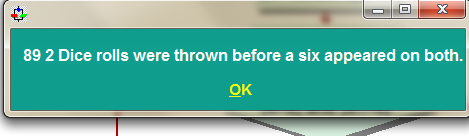
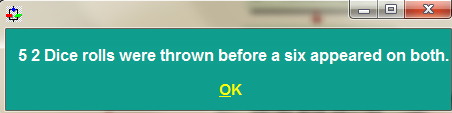




Question 5: Okay so this the one problem I have the least amount of confidence in, only because I had trouble really getting a good grip on “visual basic” from the couple of websites I found that offered programming references. This of course, falls only in the department of my own lack of discipline and dedication to web searching, but I think from what I found I understood it enough, though certainly not fully. From what I did begin to comprehend, however, highly intrigued me on overall idea of learning, not just visual basic at some point in the future, but any other programming language, just due to the fact that it simply offered a whole new range of possibilities from a different perspective. The website I used, was <http://msdn.microsoft.com/en-us/library/sh9ywfdk.aspx> . I used many links from this central page to find as much information on all of the commands I was attempting to convert. From what I understood, to convert them you simply input a value, assigning it to the variable of “mystery.” The “answer,” is the converted with a formula of /6 and add five. I got tricked up with the if statement, only because I did not read the script on the exam sheet closely enough, and originally attempted to code it for Answer < 15 and Answer mod 2=0. When I was having coding issues and getting errors on that, I re-read the problem and saw that the inputted mystery value was the variable contingent on being even or odd. Then, once the determination is made regarding whether or not these conditions are found to be that the answer is less than fifteen and the original input was even, or not, the appropriate output statement is executed. Since there was no real script on what the output statement should have been, I got super creative and just had them state their mathematical relations to the if condition.

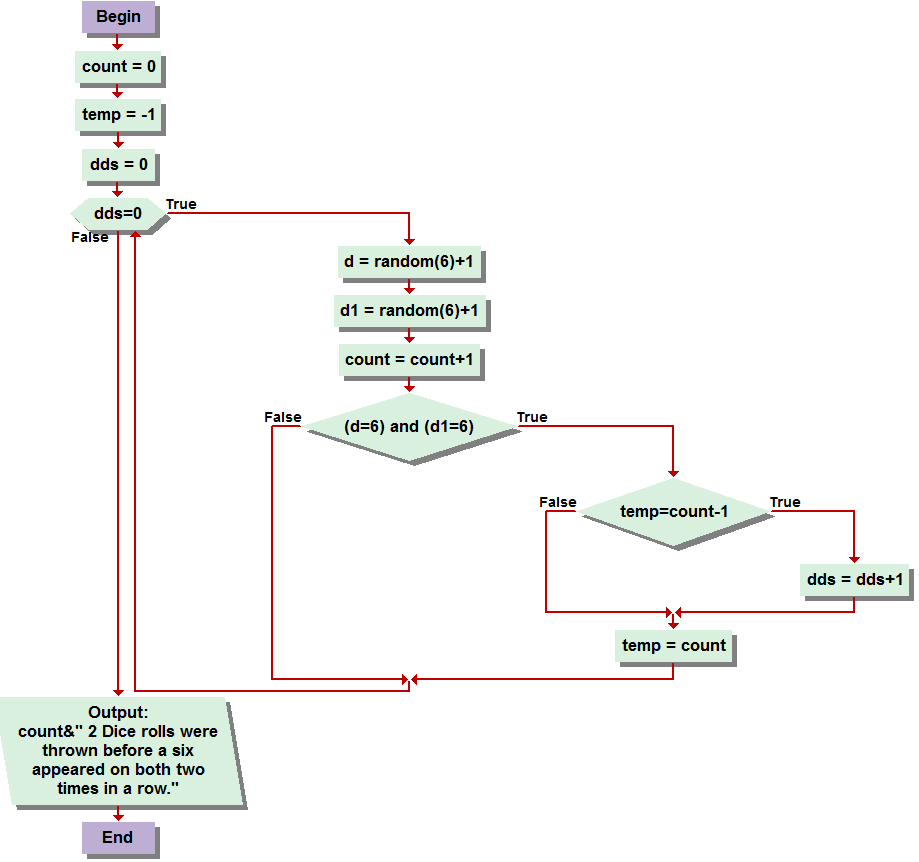
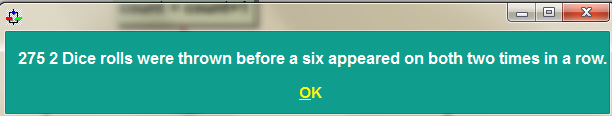
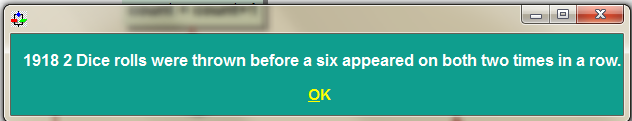
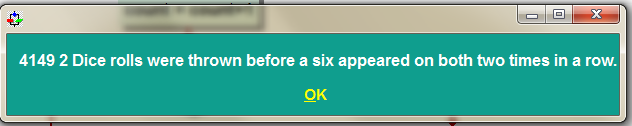


Question 6:   
Six was fun. I had wanted to do some more with dice rolling problems, especially after reading some of the practice problems at the end of the chapter in which they were introduced. So for this one, I started with a structure very similar to the example out of the book on how to formulate a dice rolling simulation. Being as that it asked for one thousand rolls, I created a for loop from one to one thousand by one. Prior to this, I should mention, I created an array with the maximum of twelve. Some of the steps I used throughout this problem, as well as in the output statements, were essentially needless for the end result of exactly what we were suppose to determine. I understand this, but as a formality, and having limited experience with the dice rolling simulation, with special regards to a two dice rolling simulation, I was simply using a ground foundation from the book to insure I got an accurate simulation to begin with. Inside the for loop, two nearly simultaneous assignments are implemented to create the dice rolling simulation. D and D1. They each generate the rolls with a random value, upper bound of 6, plus one. We then get to the heart of the problem with the oh so essential XOR condition. The statement, in broad terms, increases a counter, that I labeled “sixct” by one each time a six is rolled on only one dice, not two. I was not exactly sure if this was what was intended but from the answer I received on the message board in regards to trying to get more detailed specifications on the very first problem, and was told to use the my best judgment and to my best ability, I did just that throughout the entire exam. I realize that you may have us to only record the simulation of only one or the other dice rolling a six, but I thought this was a better, and of course, more sophisticated manor in which to answer the question. Recording all the rolls and only when one or the other, not both, equal a six, to increase the counter. The second for loop used to generate a unnecessary outputting of all the rolls, which were determined by adding the value of d and d1 together, which were then stored into the array, had nothing to do with finding out how many times a six was rolled on only one die. The final output statement displays the value of the six count with a well worded sentence illustrating and elaborating on the main focus of the assignment. I should also include that an assignment is used in the original for loop or Count(sum)=Count(sum)+1 to keep track of, again, the completely needless record of each die roll.



Question 7:  
This was a fun exciting for ground to the extra credit problem. While I found this one rather simple to code, the extra credit problem, which took much more thought, focus, and concentration, would have never been possible if not for this one. A while loop is created, continuously rolling dice until the sentinel is triggered. Prior to this, place holders for variables “Count” and “Double Sixes” or “DS” were created. As long as double sixes, DS, the sentinel value, remains at zero, dice are rolled! With the same dice rolling assignment of D and D1 = random(6)+1 generating their value, they are created and then conditioned. If they are found to BOTH have a value of 6, the sentinel is triggered killing the loop. Assuming, at least for my trials, that this never occurred on the first roll, when it doesn’t, and the If statement is found to be false, the counter variable is increased by one. Until the sentinel is met and the loop prompts, the program keeps track of every roll until then, and when it is over, displays the count value with an explanatory output statement. I included a few examples of how many times it took me to “get out of jail,” if this were monopoly. Although I am rather sure any doubles would get you out of jail, for it would be rather cruel to expect anyone to play again if they were in jail for 89 turns.

Question 8:  
Python, according to my sister, who is a computer science major at Harvard university, and a few years younger than I am, tells me python is a very fascinating programming langue, very complex, and user friendly. Although she has not learned much of yet, she has expressed her desire to do so when we spoke of the second half of this class on facebook. Exciting to me, this is of course, I have a great deal of anticipation to dive in, for while I do find visual logic quite useful for what we have learned so far, and a great stepping stone for somebody like me who has such little experience programming, I can imagine it is slightly simplistic in comparison to bigger and better programming languages. My first step into understanding how to go about learning this language is, of course, google! So I googled it, and found an even more powerful tool, wiki! <http://wiki.python.org/moin/BeginnersGuide> to be specific. From there, a slew of links to guides and tutorials are available for free, much like the software itself. The beginners guide highlights that not only is the software for engineering the programs free, but also is the exchange and customization of programs among other software developers free. It goes on to example basic features of data types, strings, module groupings, and a couple of other things I am not certain of their definition, but very eager to find out. The next step of course would be to download the software as well as a few guides to try and make head or tails of any and as much of it as I can. Although, I am hesitating on this just for another day for I am sure that some links will be made available to all of the students equally so that we can all be on the same page. Or at least, as close to as on the same page as we can be, since sometimes I feel as though I am just slightly ahead :D. Although, I am certainly not the most outstretched leader in the pack, for I have seen the work other students have done on their homework sets, and much of which is quite impressive. Regardless, so far I have felt personally to have a tight and expanding grasp on the material as it has been presented to me and hope I can continue this feeling throughout not only the rest of this course but the entire duration of my computer networking education at montco!



Extra Credit!:

So the extra credit problem rocked. Took a great deal of my own genius to accomplish, and while it Is completely dependent on random integers, therefore making it virtually impossible to test your theory on the result and expect to determine whether or not your right or wrong, you must simply trust in the process and know you got it! So I started with the program I had devised for problem seven. Although I changed a variable and added one, and at one point had another loop that I realized was not correct and toyed with it, in all, for a great deal before feeling confident, not in the solution, but even in a way to go about coding any possible solution. So we have the same double roll generator with random(6)+1 and what not, and the same if condition that would ultimately play a part in triggering the sentinel. It works, however, for this problems, with a conditional statement within a conditional statement. This is the key to the solution. So the first if statement tests if both values of the doll dice roll are sixes. If they are, great! Were on to the tricky part. When this occurs, it then tests to see if a variable “temp” is equal to count minus 1. But, temp is assigned to negative one at the beginning of the program, and only changes its value to the current count, kept track of with each dice roll generated, increasing by one in the while loop. The temp takes on the value of the count when double sixes are rolled. So if double sixes are rolled consecutively, then when the consecutive roll actually occurs, the first test determining both values to be double sixes, then leads to the test of temp. as long as temp is count-1, you are assured you have rolled them two times in a roll, and the count is displayed in an output statement reading, “count” 2 dice rolls were thrown before a six appeared on both two times in a row. And there you have it, double sixes, and my midterm! Thank you and goodnight.