Report C Assignment

Suhail Munshi. 201231965

1) For the first solution I implemented, I first used scanf to retrieve the first 3 integers and input their values into variables named row,columns and steps. Then I used scanf again to retrieve the character values and input them into an array. I used an infinite loop to verify my input and insure that no empty values or line breaks were taken as characters in the array. Next I used memcpy, a method I did not learn in lectures to copy the values of the array to a second array called newArr. Memcpy uses the format memcpy(newArr,array,size(newArr)). Where newArr is the array we are copying to and array is the array we are copying from. Using this method uses less memory then using two for loops and assigning the values one by one. Next I used 3 indented for loops in order to process the array. The outermost for loop handled each iteration of the array, while the other for loops allows us to go through each element of the array in order, from left to right. Within the inner most for loop I had an integer count set to 0. This integer counts the number of live neighbours. This is done using a total of 16 if statements. The if statements are set up so that there are 8 outer if statements, and within each one of those is another if statement. The inner if statement check the values of each of the neighbours of the item of the array with the character 'X' to determine if they are alive. If this condition is met, the integer count is incremented. The outer for loops prevent the array from going out of bounds by checking the value of the rows and columns that the iteration is on. Finally, once the count if statements have been completed, the rules of the game of life are applied by using 3 if statements to change the values of the second array. This first array is used for determining the value of count for each item, so it is not reassigned to prevent corruption of the data. Once all of the iterations are complete, the second array is sent to the print method to output the result.

2) For the second solution, I used scanf to input the first three integers. I then used scanf again and compared it with EOF, inputting each value into one of two arrays. One holds the time each car arrives the other holds the row on which the car arrives. Lastly, I used the values of rows and columns to declare a 2d array. I then set all values of the 2d array to '.' Using a function called memset. This works using the format memset(array,'.',sizeof(array)) where '.' Is limited to be a single bit character. After this, I used 3 indented for loops. The outermost for loop goes through each successive iteration. The other for loops allow us to go through each value of the array. The difference from the previous problem is that these for loops allowed us to read each element from right to left. This was done by using the inner most for loop, which was set to read backwards from the highest values until 0 is reached. The inner most for loop also contains if statements for checking whether the elements is a '1'. The program then uses if statements to determine what to do. If the 1 is at the end of the array,it is replaces with '.' Otherwise the one is swapped with the character in front. This allows the cars to move along the highway. Inside the outermost for loop,after the for loops that allow us to traverse the array I also have a while loop. This while loop checks the value

of the outermost for loop with the time that cars are supposed to arrive using the array timeAlloc[count] where count is the index mentioned earlier. Within this while loop, a ‘1’ is introduced on the row determined by the second array rowAllow with the index count as mentioned earlier. Finally, the value of count is incremented to check whether the next car is also supposed to arrive during this iteration. If the conditions are not met, the outermost for loop continues until they are. Finally, after all the for loops the 2d array is sent to the print method where the final output is produced.