Report of assignment 4

Jensen D1265209

In assignment 4-1, I define tow functions including hexadecimal to decimal and decimal to hexadecimal. In these two functions, I used character conversion by ascii. Also, there are the function, printChar and hexadecimal. Because more digits need to be stored, used unsigned long long type for hexadecimal. What I use for addition is the concept of upright addition. At the very beginning of the program, you need to determine whether the entered number satisfies the requirement that both numbers are not 0. In addition, we must first determine the length of the two values. When there are more digits in the longer one, we only need to add it to the carry of the previous digit. In the processing of carry, the answer after the two methods is divided by 16 to know how much the carry is. In the part where the answer is printed, the heavier part is to count spaces so that the calculations can be aligned.

The multiplication part also includes the addition part. However, in the multiplication program, I have also changed the way I write the addition. I no longer specifically formulate functions for converting hexadecimal to decimal and converting decimal to hexadecimal. Although I removed these two functions, I directly wrote the transformation part and used a ternary operator for the conversion. This function is designed to calculate the product of two hexadecimal numbers and return the result as a hexadecimal string. Initially, it checks if 'd' is zero. If it's not, the multiplication operation starts. Two loops are employed here - one to traverse the input hexadecimal number string 'n' and another to handle carry operations. During the multiplication process, each hexadecimal digit character is converted into its corresponding decimal value, and then the multiplication is carried out. While computing the product, potential carry values are managed. The final result is stored in the 'product' string. If there's a final carry generated during the product calculation, it's inserted at the beginning of the resulting product string. Additionally, zeros are added at the end of the product string as needed to achieve a left shift. If 'd' is zero, the result is simply zero, and "0" is stored in the 'product'. This function handles multiplication of hexadecimal numbers while considering carry operations involved in the hexadecimal arithmetic. This section of the code primarily handles the multiplication of two hexadecimal numbers and displays the process and final outcome. Initially, it checks the lengths of the two input numbers, ensuring that the larger of the two numbers is represented by `n2` and the smaller one by `n1`, facilitating the multiplication process. Next, the code initializes a string `product` to "0" and begins the multiplication. It iterates through each digit in `n2` and performs

multiplication with the corresponding digit in `n1`. After each digit multiplication, the partial product is added to `product`, resulting in the complete product at the end. Once the multiplication is done, the code formats the output differently based on whether the numbers were swapped or not. If the numbers were swapped, it prints the output in the format `n2 * n1 = product`; otherwise, it prints `n1 * n2 = product`. Finally, the code prints the final result in decimal form using the `hexadecimal` function, completing the process of multiplication in hexadecimal and displaying the outcome.