# **CS 410 Assembly to C++ Activity**

Step 1: Convert the assembly code into C++ code.

Step 2: Explain the function of the converted C++ code.

| **Assembly Code** | **C++ Code** | **Explanation of Functionality** |
| --- | --- | --- |
| movl −8(%rbp), %eax sall $3, %eax subl $3, %eax movl %eax, −4(%rbp) | int result = (input << 3) - 3; | This code takes a value from the variable input (located at -8(%rbp)), multiplies it by 8 using a left shift (<< 3), subtracts 3, and stores the result in the variable result (at -4(%rbp)). |
| movl −8(%rbp), %eax sall $2, %eax subl $1, %eax leal 7(%rax), %edx testl %eax, %eax cmovs %edx, %eax sarl $3, %eax  movl %eax, −4(%rbp) | int temp = (input << 2) - 1;  int result = (temp < 0) ? ((temp + 7) >> 3) : (temp >> 3); | This code first calculates (input \* 4) - 1. If the result is negative, it rounds the value up before right-shifting it by 3. The cmovs instruction performs conditional move if sign is set, which is similar to using the ternary operator in C++ for signed value rounding. |
| movl −8(%rbp), %eax leal 7(%rax), %edx testl %eax, %eax cmovs %edx, %eax sarl $3, %eax movl −8(%rbp), %edx sall $2, %edx addl %edx, %eax  movl %eax, −4(%rbp) | int temp = input;  int result = ((temp < 0) ? ((temp + 7) >> 3) : (temp >> 3)) + (input << 2); | This code combines two calculations:  (input + 7) >> 3 for negative numbers, or input >> 3 for non-negative numbers — a division by 8 with rounding if negative.  And:  (input << 2) — multiplies the input by 4. Then it adds the results together. |