# **CS 410 Binary to C++ Activity Template**

## **File One – assignment4\_1**

**Step 2:** Explain the functionality of the blocks of assembly code.

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| movl $0x1,-0x8(%rbp) | Initialize variable a = 1 |
| cmp $0x9,-0x8(%rbp) and jg | Outer loop from 1 to 9 |
| movl $0x1,-0xc(%rbp) | Initialize inner loop i = 1 |
| cmp $0x9,-0xc(%rbp) and jg | Inner loop from 1 to 9 |
| imul -0xc(%rbp),%eax | Calculate x = a \* i |
| callq sequences | Format and print the result |
| addl $0x1,-0xc(%rbp) | Increment inner loop counter |
| addl $0x1,-0x8(%rbp) | Increment outer loop counter |

**Step 4:** Convert the assembly code to C++ code.

**Step 5:** Explain how the C++ code performs the same tasks as the blocks of assembly code.

| **Blocks of Assembly Code** | **C++ Code** | **Explanation of Functionality** |
| --- | --- | --- |
| Loop initialization | for (a = 1; a <= 9; a++) | Sets up outer loop |
| Loop logic | for (i = 1; i <= 9; i++) | Sets up inner loop |
| Multiplication | x = a \* i; | Performs product |
| Output | cout << ... | Matches callq print chain |
| Increment | i++ and a++ | Controlled with addl |

## **File Two - assignment4\_2**

**Step 2:** Explain the functionality of the blocks of assembly code.

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| callq at 0x25 | Prompts user for input |
| callq at 0x52 | Reads integer into variable |
| imul %eax, %edx, then imul %edx, %eax | Cubes the input value |
| cvtsi2sd, mulsd | Converts to double, multiplies by 1.5 |
| callq near 0x8a and 0xa7 | Outputs result to screen |

**Step 4:** Convert the assembly code to C++ code.

**Step 5:** Explain how the C++ code performs the same tasks as the blocks of assembly code.

| **Blocks of Assembly Code** | **C++ Code** | **Explanation of Functionality** |
| --- | --- | --- |
| cin logic | cin >> input; | Reads user input |
| Cubing logic | input = input \* input; input = input \* input; | Computes input^3 |
| Convert to double | static\_cast<double>(input) | Matches cvtsi2sd |
| Multiply | result = result \* 1.5; | Matches mulsd |
| Output | cout << ... | Prints final result |

## **File Three - assignment4\_3**

**Step 2:** Explain the functionality of the blocks of assembly code.

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| movl $0x1,-0xc(%rbp) | Initializes row counter for upper half |
| cmp with N | Controls how many rows of stars are printed |
| Nested movl $0x1,-0x14 and cmp to star-count expression | Builds triangle/diamond shape |
| callq sequences | Each triggers a cout << print line with space or star |
| addl instructions | Increment loop variables |

**Step 4:** Convert the assembly code to C++ code.

**Step 5:** Explain how the C++ code performs the same tasks as the blocks of assembly code.

| **Blocks of Assembly Code** | **C++ Code** | **Explanation of Functionality** |
| --- | --- | --- |
| Input | cin >> rows; | User types the number of rows |
| Outer loops | for (i = 1; i <= rows; i++) | Controls each row |
| Inner loops | for (j = 1 to rows-i), for (k = 1 to 2\*i-1) | Prints spaces and stars |
| Print | cout << "\*" or " " | Produces pyramid/diamond visual |
| Repetition | Second for section | Creates mirrored lower half |

## **File Four - assignment4\_4**

**Step 2:** Explain the functionality of the blocks of assembly code.

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| movq and cin calls | Reads a large integer input |
| % 10, %rip magic number | Extracts rightmost digit |
| shlq | Doubles the multiplier (bit shift left) |
| imul and add | Multiplies digit by multiplier and adds to sum |
| div by 10 | Moves to next digit |
| Loop | Continues until number is 0 |
| Final callq | Prints accumulated sum |

**Step 4:** Convert the assembly code to C++ code.

**Step 5:** Explain how the C++ code performs the same tasks as the blocks of assembly code.

| **Blocks of Assembly Code** | **C++ Code** | **Explanation of Functionality** |
| --- | --- | --- |
| Read number | cin >> number; | Gets input from user |
| Loop | while (number != 0) | Loops until digits are processed |
| Digit math | digit = number % 10; | Extracts digit |
| Multiply + sum | sum += digit \* multiplier;4 | Accumulates weighted sum |
| Bit shift | multiplier <<= 1; | Doubles multiplier |
| Divide input | number /= 10; | Moves to next digit |