# CS 410 C++ to Assembly With Loops Activity Template

Step 1: Explain the functionality of the C++ code.

## C++ Code Functionality

| C++ Line of Code | Explanation of Functionality |
| --- | --- |
| #include<iostream> | includes the header file for input-output stream, allowing the use of cout and cin commands. |
| using namespace std; | using a standard namespace |
| int main() | the start of the main function's body. |
| int num, i; | Declares two integer variables num and i. |
| int product =1; | Variable product and initializes it to 1. |
| cout<<"Enter a number:\n"<< endl; | Outputs a message asking for a number, followed by a new line. |
| cin>>num; | Reads an integer from the standard input and stores it in num. |
| for(i=num;i>0; i--) | a for-loop where i starts from num and decrements until it reaches 0. |
| product = product \* i; | Inside the loop, multiplies product by i at each iteration. |
| cout<<"The factorial for " << num << "is: \n"<< product; | Outputs the computed factorial of the number. |
| return 1; | Returns 1 to indicate successful program execution. |
| } | end of the main function's body. |

Step 2: Convert the C++ file into assembly code.

Step 3: Align each line of C++ code with the corresponding blocks of assembly code.

## C++ to Assembly Alignment

| C++ Line of Code | Blocks of Assembly Code |
| --- | --- |
| int num, i; | .LFB1493: |
| int product =1; | .cfi\_startproc  pushq %rbp  .cfi\_def\_cfa\_offset 16  .cfi\_offset 6, -16  movq %rsp, %rbp  .cfi\_def\_cfa\_register 6  subq $32, %rsp  movq %fs:40, %rax  movq %rax, -8(%rbp)  xorl %eax, %eax  movl $1, -12(%rbp)  leaq .LC0(%rip), %rsi  leaq \_ZSt4cout(%rip), %rdi |
| cout<<"Enter a number:\n"<< endl; | .LC0:  .string "Enter a number:\n" |
| cin>>num; | movq %rax, %rsi  leaq \_ZSt3cin(%rip), %rdi  call \_ZNSirsERi@PLT  movl -20(%rbp), %eax  movl %eax, -16(%rbp) |
| for(i=num;i>0; i--) | movl -12(%rbp), %eax  movl %eax, %esi  movq %rdx, %rdi  call \_ZNSolsEi@PLT  movl $1, %eax  movq -8(%rbp), %rcx  xorq %fs:40, %rcx |
| product = product \* i; | movl -12(%rbp), %eax  movl %eax, %esi  movq %rdx, %rdi  call \_ZNSolsEi@PLT  movl $1, %eax  movq -8(%rbp), %rcx  xorq %fs:40, %rcx |
| cout<<"The factorial for " << num << "is: \n"<< product; | .LC1:  .string "The factorial for "  .LC2:  .string "is: \n" |
| return 1; | .L8:  nop  leave  .cfi\_def\_cfa 7, 8  ret  .cfi\_endproc |

Step 4: Explain how the blocks of assembly code perform the same tasks as the C++ code.

## Assembly Functionality

| Blocks of Assembly Code | Explanation of Functionality |
| --- | --- |
| .LFB1493: | Function begin label; used for debugging and linking. |
| .cfi\_startproc  pushq %rbp  .cfi\_def\_cfa\_offset 16  .cfi\_offset 6, -16  movq %rsp, %rbp  .cfi\_def\_cfa\_register 6  subq $32, %rsp  movq %fs:40, %rax  movq %rax, -8(%rbp)  xorl %eax, %eax  movl $1, -12(%rbp)  leaq .LC0(%rip), %rsi  leaq \_ZSt4cout(%rip), %rdi | stack frame setup, and loading string and cout object addresses. |
| .LC0:  .string "Enter a number:\n" | String literal for output. |
| movq %rax, %rsi  leaq \_ZSt3cin(%rip), %rdi  call \_ZNSirsERi@PLT  movl -20(%rbp), %eax  movl %eax, -16(%rbp) | Reading an integer from cin and storing it. |
| movl -12(%rbp), %eax  movl %eax, %esi  movq %rdx, %rdi  call \_ZNSolsEi@PLT  movl $1, %eax  movq -8(%rbp), %rcx  xorq %fs:40, %rcx | Outputting an integer, and stack canary verification. |
| movl -12(%rbp), %eax  movl %eax, %esi  movq %rdx, %rdi  call \_ZNSolsEi@PLT  movl $1, %eax  movq -8(%rbp), %rcx  xorq %fs:40, %rcx | Outputting an integer, and stack canary verification. |
| .LC1:  .string "The factorial for "  .LC2:  .string "is: \n" | String literals for output. |
| .L8:  nop  leave  .cfi\_def\_cfa 7, 8  ret  .cfi\_endproc | . Function epilogue and return |