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

## **File One**

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

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| movl $0x1,-0x8(%rbp)  cmpl $0x9,-0x8(%rbp)  jg a3 <main+0xa3> | Moves 4 bytes of $0x1 and assigns into - 0x8(%rbp)  Compares $0x9 and -0x8(%rbp)  Jumps to a3 <main+0xa3> IF greater |
| cmpl $0x9,-0xc(%rbp)  jg 9a4 <main+0x9a> | Compares $0x9 and -0xc(%rbp)  Jumps to 9a4 <main+0x9a> IF greater |
| mov -0x8(%rbp),%eax  imul -0xc(%rbp),%eax  mov %eax,-0x4(%rbp) | Moves and Assigns -0x8(%rbp) into %eax  Multilply -0xc(%rbp) by %eax  Moves and Assigns %eax into -0x4(%rbp) |
| mov -0x8(%rbp),%eax  mov %eax,%esi  lea 0x2006da(%rip),%rdi  callq 7e0 <\_ZNSolsEi@plt>  lea 0x153(%rip),%rsi  mov %rax,%rdi  callq 7b0 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 7e0 <\_ZNSolsEi@plt>  lea 0x138(%rip),%rsi  mov %rax,%rdi  callq 7b0 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov -0x4(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 7e0 <\_ZNSolsEi@plt>  mov %rax,%rdx  mov 0x20063d(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 7c0 <\_ZNSolsEPFRSoS\_E@plt> | Moves and Assigns -0x8(%rbp) into %eax  Moves and Assigns %eax into %esi  Reads 0x2006da(%rip) and loads into %rdi  Calls and outputs 7e0 <\_ZNSolsEi@plt>  Reads 0x153(%rip) and loads into %rsi  Moves and Assigns %rax into %rdi  Calls and outputs 7b0 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  Moves and Assigns %rax into %rdx  Moves and Assigns -0xc(%rbp) into %eax  Moves and Assigns %eax into %esi  Moves and Assigns %rdx into %rdi  Calls and outputs 7e0 <\_ZNSolsEi@plt>  Reads 0x138(%rip) and loads into %rsi  Moves and Assigns %rax into %rdi  Calls and outputs 7b0 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  Moves and Assigns %rax into %rdx  Moves and Assigns -0x4(%rbp) into %eax  Moves and Assigns %eax, into %esi  Moves and Assigns %rdx into %rdi  Calls and outputs 7e0 <\_ZNSolsEi@plt>  Moves and Assigns %rax into %rdx  Moves and Assigns 0x20063d(%rip) into %rax  Moves and Assigns %rax into %rsi  Moves and Assigns %rdx into %rdi  Calls and outputs 7c0 <\_ZNSolsEPFRSoS\_E@plt> |
| mov    $0x0,%eax  leaveq  retq | Moves and Assigns $0x0 into %eax  Cleans stack and prepares exit  Return and continue execution |

**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** |
| --- | --- | --- |
| movl $0x1,-0x8(%rbp)  cmpl $0x9,-0x8(%rbp)  jg a3 <main+0xa3>  addl   $0x1,-0x8(%rbp)  jmpq   919 <main+0xf> | for (a = 1; i <= 9; a++) | 1 is moved into the register and is compared to 9. If true, continues loop until equal to 9. |
| cmpl $0x9,-0xc(%rbp)  jg 9a4 <main+0x9a>   mov -0x8(%rbp),%eax  imul -0xc(%rbp),%eax  mov %eax,-0x4(%rbp) | for (i = 1; i <= 9; i++){  x = a \* i; | 1 is moved into the register and is compared to 9. If true, continue loop until equal to 9. Then multiplies values to solve for x. |
| mov -0x8(%rbp),%eax  mov %eax,%esi  lea 0x2006da(%rip),%rdi  callq 7e0 <\_ZNSolsEi@plt>  lea 0x153(%rip),%rsi  mov %rax,%rdi  callq 7b0 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 7e0 <\_ZNSolsEi@plt>  lea 0x138(%rip),%rsi  mov %rax,%rdi  callq 7b0 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov -0x4(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 7e0 <\_ZNSolsEi@plt>  mov %rax,%rdx  mov 0x20063d(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 7c0 <\_ZNSolsEPFRSoS\_E@plt> | cout << a << " \* " << i << " = " << x << endl; | Prints the output |
| mov    $0x0,%eax  leaveq  retq | return 0 | Return 0 and successfully exits program |

## **File Two**

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

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| mov %fs:0x28,%rax  mov %rax,-0x8(%rbp)  xor %eax,%eax  lea 0x191(%rip),%rsi  lea 0x201601(%rip),%rdi  callq 890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov 0x2015a2(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi | Moves and Assigns %fs:0x28 into %rax  Moves and Assigns %rax into -0x8(%rbp)  Compares %eax and generates first output  Reads 0x191(%rip) and loads into %rsi  Reads 0x201601 (%rip) and loads into %rsi  Calls and outputs <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  Moves and Assigns %rax into %rdx  Moves and Assigns 0x2015a2(%rip) into %rax  Moves and Assigns %rax into %rsi  Moves and Assigns %rdx into %rdi |
| callq 8a0 <\_ZNSolsEPFRSoS\_E@plt>  lea -0x14(%rbp),%rax  mov %rax,%rsi  lea 0x2016f9(%rip),%rdi  callq 870 <\_ZNSirsERi@plt>  mov -0x14(%rbp),%edx  mov -0x14(%rbp),%eax  imul %eax,%edx  mov -0x14(%rbp),%eax  imul %edx,%eax  mov %eax,-0x14(%rbp)  mov -0x14(%rbp),%eax  cvtsi2sd %eax,%xmm0  movsd 0x15b(%rip),%xmm1  mulsd %xmm1,%xmm0  movsd %xmm0,-0x10(%rbp)  lea 0x13a(%rip),%rsi  lea 0x20159c(%rip),%rdi  callq 890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> | Calls and outputs 8a0 <\_ZNSolsEPFRSoS\_E@plt>  Reads -0x14(%rbp) and loads into %rax  Moves and Assigns %rax into %rsi  Reads 0x2016f9(%rip) and loads into %rdi  Calls and outputs 870 <\_ZNSirsERi@plt>  Moves and Assigns -0x14(%rbp) into %edx  Moves and Assigns -0x14(%rbp) into %eax  Multilply %eax by %edx  Moves and Assigns -0x14(%rbp) into %eax  Multilply %edx by %eax  Moves and Assigns %eax into -0x14(%rbp)  Moves and Assigns -0x14(%rbp) into %eax  Moves floating-point value of 0x15b(%rip) to %xmm1  Multiples floating-point value of %xmm1 by %xmm0  Moves floating-point value of %xmm0 to -0x10(%rbp)  Reads 0x13a(%rip) and loads into %rsi  Reads 0x20159c(%rip) and loads into %rsi  Calls and outputs 890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| mov %rax,%rdx  mov -0x10(%rbp),%rax  mov %rax,-0x28(%rbp)  movsd -0x28(%rbp),%xmm0  mov %rdx,%rdi  callq 8d0 <\_ZNSolsEd@plt>  mov $0x0,%eax  mov -0x8(%rbp),%rcx  xor %fs:0x28,%rcx | Moves and Assigns %rax into %rdx  Moves and Assigns -0x10(%rbp) into %rax  Moves and Assigns %rax into -0x28(%rbp)  Multiples floating-point value of %xmm1 by %xmm0  Moves and Assigns %rdx into %rdi  Calls and outputs 8d0 <\_ZNSolsEd@plt>  Moves and Assigns $0x0 into %eax  Moves and Assigns -0x8(%rbp) into %rcx  Compares %fs:0x28 with %rcx and generates first output |
| je aba <main+0xc0>  callq 8b0 <\_\_stack\_chk\_fail@plt>  leaveq  retq | IF value is equal, jump to aba <main+0xc0>  Calls and outputs 8b0 <\_\_stack\_chk\_fail@plt>  Cleans stack and prepares exit  Return and continue execution |

**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** |
| --- | --- | --- |
| mov %fs:0x28,%rax  mov %rax,-0x8(%rbp)  xor %eax,%eax  lea 0x191(%rip),%rsi  lea 0x201601(%rip),%rdi  callq 890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov 0x2015a2(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi | float volume, pi = 3.14159;  int radius;  cout<<” Enter radius: “<< endl;  cin>>; | Declares value for volume and moves into register.  System outputs request for radius.  Assigns user input |
| callq 8a0 <\_ZNSolsEPFRSoS\_E@plt>  lea -0x14(%rbp),%rax  mov %rax,%rsi  lea 0x2016f9(%rip),%rdi  callq 870 <\_ZNSirsERi@plt>  mov -0x14(%rbp),%edx  mov -0x14(%rbp),%eax  imul %eax,%edx  mov -0x14(%rbp),%eax  imul %edx,%eax  mov %eax,-0x14(%rbp)  mov -0x14(%rbp),%eax  cvtsi2sd %eax,%xmm0  movsd 0x15b(%rip),%xmm1  mulsd %xmm1,%xmm0  movsd %xmm0,-0x10(%rbp)  lea 0x13a(%rip),%rsi  lea 0x20159c(%rip),%rdi  callq 890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> | volume = pi \* radius \* radius \* radius | Defines and calculates volume by multiplying the equation |
| mov %rax,%rdx  mov -0x10(%rbp),%rax  mov %rax,-0x28(%rbp)  movsd -0x28(%rbp),%xmm0  mov %rdx,%rdi  callq 8d0 <\_ZNSolsEd@plt>  mov $0x0,%eax  mov -0x8(%rbp),%rcx  xor %fs:0x28,%rcx | cout << "The volume is: " << volume << endl; | Outputs volume calculation |
| je aba <main+0xc0>  callq 8b0 <\_\_stack\_chk\_fail@plt>  leaveq  retq | return 0; | Return 0 and successfully exits program |

## **File Three**

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

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| mov %fs:0x28,%rax  mov %rax,-0x8(%rbp)  xor %eax,%eax  movl $0x1,-0xc(%rbp)  lea 0x256(%rip),%rsi  lea 0x20163a(%rip),%rdi  callq 860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov 0x2015db(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 870 <\_ZNSolsEPFRSoS\_E@plt>  lea -0x18(%rbp),%rax  mov %rax,%rsi  lea 0x201732(%rip),%rdi  callq 840 <\_ZNSirsERi@plt>  mov -0x18(%rbp),%eax  sub $0x1,%eax  mov %eax,-0xc(%rbp) | Moves and Assigns %fs:0x28 into %rax  Moves and Assigns %rax into -0x8(%rbp)  Compares %eax and generates first output  Moves 4 bytes of $0x1 into -0xc(%rbp)  Reads 0x256 (%rip) and loads into %rsi  Reads 0x20163a (%rip) and loads into %rsi  Calls and outputs 860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  Moves and Assigns %rax into %rdx  Moves and Assigns 0x2015db(%rip) into %rax  Moves and Assigns %rax into %rsi  Moves and Assigns %rdx into %rdi  Calls and outputs 870 <\_ZNSolsEPFRSoS\_E@plt>  Reads -0x18(%rbp) and loads into %rax  Moves and Assigns %rax into %rsi  Reads 0x201732(%rip) and loads into %rsi  Calls and outputs 840 <\_ZNSirsERi@plt>  Moves and Assigns -0x18(%rbp) into %eax  Subtract $0x1 from %eax  Moves and Assigns %eax into -0xc(%rbp) |
| movl $0x1,-0x10(%rbp)  mov -0x18(%rbp),%eax  cmp %eax,-0x10(%rbp)  jg a9d <main+0xe3>  movl $0x1,-0x14(%rbp)  mov -0x14(%rbp),%eax  cmp -0xc(%rbp),%eax  jg a53 <main+0x99>  lea 0x209(%rip),%rsi  lea 0x2015d8(%rip),%rdi  callq 860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  addl $0x1,-0x14(%rbp)  jmp a32 <main+0x78>  subl $0x1,-0xc(%rbp)  movl $0x1,-0x14(%rbp)  mov -0x10(%rbp),%eax  add %eax,%eax  sub $0x1,%eax  cmp %eax,-0x14(%rbp)  jg a84 <main+0xca>  lea 0x1da(%rip),%rsi  lea 0x2015a7(%rip),%rdi  callq 860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  addl $0x1,-0x14(%rbp)  jmp a5e <main+0xa4>  lea 0x1c3(%rip),%rsi  lea 0x20158e(%rip),%rdi  callq 860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  addl $0x1,-0x10(%rbp)  jmp a23 <main+0x69>  movl $0x1,-0xc(%rbp)  movl $0x1,-0x10(%rbp)  mov -0x18(%rbp),%eax  sub $0x1,%eax  cmp %eax,-0x10(%rbp)  jg b2b <main+0x171>  movl $0x1,-0x14(%rbp)  mov -0x14(%rbp),%eax  cmp -0xc(%rbp),%eax  jg ade <main+0x124>  lea 0x17e(%rip),%rsi  lea 0x20154d(%rip),%rdi  callq 860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> | Moves 4 bytes of $0x1 into -0x10(%rbp)  Moves and Assigns -0x18(%rbp) into %eax  Compares %eax and -0x10(%rbp)  Jumps to a9d <main+0xe3> IF greater  Moves 4 bytes of $0x1 into -0x14(%rbp)  Moves and Assigns -0x14(%rbp) into %eax  Compares -0xc(%rbp) and %eax  Jumps to a53 <main+0x99> IF greater  Reads 0x209 (%rip) and loads into %rsi  Reads 0x2015d8 (%rip) and loads into %rsi  Calls and outputs 860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  Adds $0x1 and -0x14(%rbp)  Jump to a32 <main+0x78> function  Subtracts $0x1,-0xc(%rbp)  Moves 4 bytes of $0x1 into -0x14(%rbp)  Moves and Assigns -0x10(%rbp) into %eax  Adds %eax and %eax  Subtracts $0x1 by %eax  Compares %eax and -0x14(%rbp)  Jumps to a84 <main+0xca>  Reads 0x1da(%rip) and loads into %rsi  Reads 0x2015a7 (%rip) and loads into %rsi  Calls and outputs 860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  Adds $0x1 and -0x14(%rbp)  Jumps to a5e <main+0xa4>  Reads 0x1c3(%rip) and loads into %rsi  Reads 0x20158e(%rip) and loads into %rdi  Calls and outputs 860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  Adds $0x1 and -0x10(%rbp)  Jump to a23 <main+0x69>  Moves 4 bytes of $0x1 into -0xc(%rbp)  Moves 4 bytes of $0x1 into -0x10(%rbp)  Moves and Assigns -0x18(%rbp) into %eax  Subtract $0x1 by %eax  Compares %eax and -0x10(%rbp)  Jumps to b2b <main+0x171>  Moves 4 bytes of $0x1 into -0x14(%rbp)  Moves and Assigns -0x14(%rbp) into %eax  Compares -0xc(%rbp) and %eax  Jumps to ade <main+0x124>  Reads 0x17e(%rip) and loads into %rsi  Reads 0x20154d(%rip) and loads into %rdi  Calls and outputs 860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| addl $0x1,-0x14(%rbp)  jmp abd <main+0x103>  addl $0x1,-0xc(%rbp)  movl $0x1,-0x14(%rbp)  mov -0x18(%rbp),%eax  sub -0x10(%rbp),%eax  add %eax,%eax  sub $0x1,%eax  cmp %eax,-0x14(%rbp)  jg b12 <main+0x158>  lea 0x14c(%rip),%rsi  lea 0x201519(%rip),%rdi  callq 860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  addl $0x1,-0x14(%rbp)  jmp ae9 <main+0x12f>  lea 0x135(%rip),%rsi  lea 0x201500(%rip),%rdi  callq 860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  addl $0x1,-0x10(%rbp)  jmp aab <main+0xf1>  mov $0x1,%eax  mov -0x8(%rbp),%rcx  xor %fs:0x28,%rcx  je b44 <main+0x18a>  callq 880 <\_\_stack\_chk\_fail@plt>  leaveq  retq | Adds $0x1 and -0x14(%rbp)  Jumps to abd <main+0x103>  Adds $0x1 and -0xc(%rbp)  Moves 4 bytes of $0x1 into -0x14 (%rbp)  Moves and Assigns -0x18(%rbp) into %eax  Subtracts -0x10(%rbp) from %eax  Adds %eax and %eax  Subtracts $0x1 by %eax  Compares %eax and -0x14(%rbp)  Jumps to b12 <main+0x158>  Reads 0x14c(%rip) and loads into %rsi  Reads 0x201519(%rip) and loads into %rdi  Calls and outputs 860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  Adds $0x1 and -0x14(%rbp)  Jumps to ae9 <main+0x12f>  Reads 0x135(%rip) and loads into %rsi  Reads 0x201500(%rip) and loads into %rdi  Calls and outputs 860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  Adds $0x1 and -0x10(%rbp)  Jumps to aab <main+0xf1>  Moves and Assigns $0x1 into %eax  Moves and Assigns -0x8(%rbp) into %rcx  Compares %fs:0x28 and %rcx  Jumps to b44 <main+0x18a>  Calls and outputs 880 <\_\_stack\_chk\_fail@plt>  Cleans stack and prepares exit  Return and continue execution |
|  |  |
|  |  |

**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** |
| --- | --- | --- |
| mov %fs:0x28,%rax  mov %rax,-0x8(%rbp)  xor %eax,%eax  movl $0x1,-0xc(%rbp)  lea 0x256(%rip),%rsi  lea 0x20163a(%rip),%rdi  callq 860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov 0x2015db(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 870 <\_ZNSolsEPFRSoS\_E@plt>  lea -0x18(%rbp),%rax  mov %rax,%rsi  lea 0x201732(%rip),%rdi  callq 840 <\_ZNSirsERi@plt>  mov -0x18(%rbp),%eax  sub $0x1,%eax  mov %eax,-0xc(%rbp) | Int rows;  cout << " Enter number of  rows\n " << endl;  cin >> rows; | Declaring variables  Outputs string  Moves value of %rax into %rsi that becomes the pending variable of user input. |
| movl $0x1,-0x10(%rbp)  mov -0x18(%rbp),%eax  cmp %eax,-0x10(%rbp)  jg a9d <main+0xe3>  movl $0x1,-0x14(%rbp)  mov -0x14(%rbp),%eax  cmp -0xc(%rbp),%eax  jg a53 <main+0x99>  lea 0x209(%rip),%rsi  lea 0x2015d8(%rip),%rdi  callq 860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  addl $0x1,-0x14(%rbp)  jmp a32 <main+0x78>  subl $0x1,-0xc(%rbp)  movl $0x1,-0x14(%rbp)  mov -0x10(%rbp),%eax  add %eax,%eax  sub $0x1,%eax  cmp %eax,-0x14(%rbp)  jg a84 <main+0xca>  lea 0x1da(%rip),%rsi  lea 0x2015a7(%rip),%rdi  callq 860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  addl $0x1,-0x14(%rbp)  jmp a5e <main+0xa4>  lea 0x1c3(%rip),%rsi  lea 0x20158e(%rip),%rdi  callq 860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  addl $0x1,-0x10(%rbp)  jmp a23 <main+0x69>  movl $0x1,-0xc(%rbp)  movl $0x1,-0x10(%rbp)  mov -0x18(%rbp),%eax  sub $0x1,%eax  cmp %eax,-0x10(%rbp)  jg b2b <main+0x171>  movl $0x1,-0x14(%rbp)  mov -0x14(%rbp),%eax  cmp -0xc(%rbp),%eax  jg ade <main+0x124>  lea 0x17e(%rip),%rsi  lea 0x20154d(%rip),%rdi  callq 860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> | for (int i = 1; i <= rows; i++)  {  for (int j = 1; j <= rows - i; j++)  {  cout << " ";  }  for (int k = 1; k <= 2 \* i - 1; k++)  {  cout << "\*";  }  cout << endl;  } | Loop states if both values are compared and true, the loop will continue to compare until false. If that statement is false, it will continue into the next loop.  The second loop will have the same concept of being compared until false. Once the statement is false, it will reach the next loop.  The third loop will do a conditional test to compare the variables. Once the loop is deemed false the loop will end.  Once all loops are complete, it will print the output. |
| addl $0x1,-0x14(%rbp)  jmp abd <main+0x103>  addl $0x1,-0xc(%rbp)  movl $0x1,-0x14(%rbp)  mov -0x18(%rbp),%eax  sub -0x10(%rbp),%eax  add %eax,%eax  sub $0x1,%eax  cmp %eax,-0x14(%rbp)  jg b12 <main+0x158>  lea 0x14c(%rip),%rsi  lea 0x201519(%rip),%rdi  callq 860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  addl $0x1,-0x14(%rbp)  jmp ae9 <main+0x12f>  lea 0x135(%rip),%rsi  lea 0x201500(%rip),%rdi  callq 860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  addl $0x1,-0x10(%rbp)  jmp aab <main+0xf1>  mov $0x1,%eax  mov -0x8(%rbp),%rcx  xor %fs:0x28,%rcx  je b44 <main+0x18a>  callq 880 <\_\_stack\_chk\_fail@plt>  leaveq  retq | for(j=1;j<=2\*i-1;j++)  cout<<"\*";  cout<<endl;  }  for(i=r-1;i>=1;i--)  {  for(j=1;j<=r-i;j++)  cout<<" ";  for(j=1;j<=2\*i-1;j++)  cout<<"\*";  cout<<endl;  }  } |  |
|  |  |  |
|  |  |  |

## **File Four**

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

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  sub $0x30,%rsp  mov %fs:0x28,%rax | Registers %rbp as base  Moves and Assigns %rsp into %rbp  Subtracts $0x20 from %rs  Moves and Assigns %fs:0x28 into %rax |
| mov %rax,-0x8(%rbp)  xor %eax,%eax  movq $0x0,-0x20(%rbp)  movq $0x1,-0x18(%rbp)  lea 0x201(%rip),%rsi  lea 0x2015f1(%rip),%rdi  callq 890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov 0x201592(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 8a0 <\_ZNSolsEPFRSoS\_E@plt>  lea -0x28(%rbp),%rax  mov %rax,%rsi  lea 0x2016e9(%rip),%rdi  callq 870 <\_ZNSirsERl@plt>  mov -0x28(%rbp),%rax  test %rax,%rax  je aec <main+0xf2>  mov -0x28(%rbp),%rcx  movabs $0x6666666666666667,%rdx  mov %rcx,%rax  imul %rdx  sar $0x2,%rdx  mov %rcx,%rax  sar $0x3f,%rax | Moves and Assigns %rax into -0x8(%rbp)  Compares %eax and generates first output  Moves 4 bytes of $0x0 into -0x20(%rbp)  Moves 4 bytes of $0x1 into -0x18(%rbp)  Reads 0x201(%rip) and loads into %rsi  Reads 0x2015f1(%rip) and loads into %rdi  Calls and outputs 890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  Moves and Assigns %rax into %rdx  Moves and Assigns 0x201592(%rip) into %rax  Moves and Assigns %rax into %rsi  Moves and Assigns %rdx into %rdi  Calls and outputs 8a0 <\_ZNSolsEPFRSoS\_E@plt>  Reads 0x28(%rbp) and loads into %rax  Moves and Assigns %rax into %rsi  Reads 0x2016e9(%rip) and loads into %rdi  Calls and outputs 870 <\_ZNSirsERl@plt>  Moves and Assigns -0x28(%rbp) into %rax  Tests %rax and %rax  Jumps to aec <main+0xf2>  Moves and Assigns -0x28(%rbp) into %rcx  Move abs value 6666666666666667 in %rdx  Moves and Assigns %eax into %esi  Multiplies %rdx  Shift %rdx to the right by $0x2  Moves and Assigns %rcx into %rax  Shift %rax to the right by $0x3 |
| sub %rax,%rdx  mov %rdx,%rax  mov %rax,-0x10(%rbp)  mov -0x10(%rbp),%rdx  mov %rdx,%rax  shl $0x2,%rax  add %rdx,%rax  add %rax,%rax  sub %rax,%rcx  mov %rcx,%rax  mov %rax,-0x10(%rbp)  mov -0x10(%rbp),%rax  imul -0x18(%rbp),%rax  add %rax,-0x20(%rbp)  shlq -0x18(%rbp)  mov -0x28(%rbp),%rcx  movabs $0x6666666666666667,%rdx  mov %rcx,%rax  imul %rdx  sar $0x2,%rdx  mov %rcx,%rax  sar $0x3f,%rax  sub %rax,%rdx  mov %rdx,%rax  mov %rax,-0x28(%rbp)  jmpq a5c <main+0x62>  lea 0x155(%rip),%rsi  lea 0x201526(%rip),%rdi  callq 890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov -0x20(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 8d0 <\_ZNSolsEl@plt>  mov %rax,%rdx  mov 0x2014b5(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 8a0 <\_ZNSolsEPFRSoS\_E@plt>  mov $0x0,%eax  mov -0x8(%rbp),%rsi  xor %fs:0x28,%rsi  je b3f <main+0x145>  callq 8b0 <\_\_stack\_chk\_fail@plt>  leaveq  retq | Subtracts %rax from %rdx  Moves and Assigns %rdx into %rax  Moves and Assigns %rax into -0x10(%rbp)  Moves and Assigns -0x10(%rbp) into %rdx  Moves and Assigns -0x10(%rbp) into %rdx  Shift %rax left by $0x2  Adds %rdx and %rax  Adds %rax and %rax  Subtracts %rax from %rcx  Moves and Assigns %rcx into %rax  Moves and Assigns %rax into -0x10(%rbp)  Moves and Assigns -0x10(%rbp) into %rax  Multiplies -0x18(%rbp) by %rax  Add %rax and -0x20(%rbp)  Shift %rbp left by 24 bytes  Moves and Assigns %eax into %esi  Move abs value 6666666666666667 in %rdx  Multiplies %rcx and %rax  Moves and Assigns %rcx into %rax  Shift %rdx to the right by $0x2  Moves and Assigns %rcx into %rax  Shift %rdx to the right by $0x2  Subtract %rax from %rdx  Moves and Assigns %rdx into %rax  Moves and Assigns %rax into -0x28(%rbp)  Jumps to a5c <main+0x62>  Reads 0x155 (%rip) and loads into %rsi  Reads 0x201526(%rip) and loads into %rdi  Calls and outputs 890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  Moves and Assigns %rax into %rdx  Moves and Assigns -0x20(%rbp) into %rax  Moves and Assigns %rax into %rsi  Moves and Assigns %rdx into %rdi  Calls and outputs 8d0 <\_ZNSolsEl@plt>  Moves and Assigns %rax into %rdx  Moves and Assigns 0x2014b5(%rip) into %rax  Moves and Assigns %rax into %rsi  Moves and Assigns %rdx into %rdi  Calls and outputs 8a0 <\_ZNSolsEPFRSoS\_E@plt>  Moves and Assigns $0x0 into %eax  Moves and Assigns -0x8(%rbp) into %rsi  Compares %fs:0x28 and %rsi  Jumps to b3f <main+0x145>  Calls and outputs 8b0 <\_\_stack\_chk\_fail@plt>  Cleans stack and prepares exit  Return and continue execution |
|  |  |
|  |  |

**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** |
| --- | --- | --- |
| mov %rax,-0x8(%rbp)  xor %eax,%eax  movq $0x0,-0x20(%rbp)  movq $0x1,-0x18(%rbp)  lea 0x201(%rip),%rsi  lea 0x2015f1(%rip),%rdi  callq 890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov 0x201592(%rip),%rax <\_ZSt4endlIcSt11char\_traitsIcEERSt13basic\_ostreamIT\_T0\_ES6\_@GLIBCXX\_3.4>  mov %rax,%rsi  mov %rdx,%rdi  callq 8a0 <\_ZNSolsEPFRSoS\_E@plt>  lea -0x28(%rbp),%rax  mov %rax,%rsi  lea 0x2016e9(%rip),%rdi  callq 870 <\_ZNSirsERl@plt>  mov -0x28(%rbp),%rax  test %rax,%rax  je aec <main+0xf2>  mov -0x28(%rbp),%rcx  movabs $0x6666666666666667,%rdx  mov %rcx,%rax  imul %rdx  sar $0x2,%rdx  mov %rcx,%rax  sar $0x3f,%rax | {  long int num, hex = 0, i = 1, remainder.  cout << "Enter binary number:" << endl;  cin >> num;  while (num!= 0)  {  remainder = num % 10;  hex = hex + remainder \* i; i = i \* 2;  num = num / 10;  }  cout << "Hexadecimal value: " << " " << hex << endl;  return 0;  } | Declares the variable of the binary number and the hexadecimal.  System outputs string and user input.  Creates a loop when the binary number is not equivalent to 0 it will jump to calculate the hexadecimal number and the remainder.  System outputs the hexadecimal number and will return to exit program. |
| sub %rax,%rdx  mov %rdx,%rax  mov %rax,-0x10(%rbp)  mov -0x10(%rbp),%rdx  mov %rdx,%rax  shl $0x2,%rax  add %rdx,%rax  add %rax,%rax  sub %rax,%rcx  mov %rcx,%rax  mov %rax,-0x10(%rbp)  mov -0x10(%rbp),%rax  imul -0x18(%rbp),%rax  add %rax,-0x20(%rbp)  shlq -0x18(%rbp)  mov -0x28(%rbp),%rcx  movabs $0x6666666666666667,%rdx  mov %rcx,%rax  imul %rdx  sar $0x2,%rdx  mov %rcx,%rax  sar $0x3f,%rax  sub %rax,%rdx  mov %rdx,%rax  mov %rax,-0x28(%rbp)  jmpq a5c <main+0x62>  lea 0x155(%rip),%rsi  lea 0x201526(%rip),%rdi  callq 890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov -0x20(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 8d0 <\_ZNSolsEl@plt>  mov %rax,%rdx  mov 0x2014b5(%rip),%rax  <\_ZSt4endlIcSt11char\_traitsIcEERSt13basic\_ostreamIT\_T0\_ES6\_@GLIBCXX\_3.4>  mov %rax,%rsi  mov %rdx,%rdi  callq 8a0 <\_ZNSolsEPFRSoS\_E@plt>  mov $0x0,%eax  mov -0x8(%rbp),%rsi  xor %fs:0x28,%rsi  je b3f <main+0x145>  callq 8b0 <\_\_stack\_chk\_fail@plt>  leaveq  retq |  |  |
|  |  |  |
|  |  |  |