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

## **File One (assignment4\_1\_fixed)**

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

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| <+0>: push %rbp  <+1>: mov %rsp, %rbp  <+4>: sub $0x10, %rsp  <+8>: movl $0x1, -0x8(%rbp)  <+15>: cmpl $0x9, -0x8(%rbp)  <+19>: jg 0x9ad <main+163>  <+25>: movl $0x1,-0xc(%rbp)  <+32>: cmpl $0x9,-0xc(%rbp)  <+36>: jg 0x9a4 <main+154>  <+38>: mov -0x8(%rbp),%eax  <+41>: imul -0xc(%rbp),%eax  <+45>: mov %eax,-0x4(%rbp)  <+48>: mov -0x8(%rbp),%eax  <+51>: mov %eax,%esi  <+53>: lea 0x2006da(%rip),%rdi  # 0x201020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+60>: callq 0x7e0 <\_ZNSolsEi@plt>  <+65>: lea 0x153(%rip),%rsi # 0xaa5  <+72>: mov %rax,%rdi  <+75>: callq 0x7b0 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+80>: mov %rax,%rdx  <+83>: mov -0xc(%rbp),%eax  <+86>: mov %eax,%esi  <+88>: mov %rdx,%rdi  <+91>: callq 0x7e0 <\_ZNSolsEi@plt>  <+96>: lea 0x138(%rip),%rsi # 0xaa9  <+103>: mov %rax,%rdi  <+106>: callq 0x7b0 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+111>: mov %rax,%rdx  <+114>: mov -0x4(%rbp),%eax  <+117>: mov %eax,%esi  <+119>: mov %rdx,%rdi  <+122>: callq 0x7e0 <\_ZNSolsEi@plt>  <+127>: mov %rax,%rdx  <+130>: mov 0x20063d(%rip),%rax # 0x200fd0  <+137>: mov %rax,%rsi  <+140>: mov %rdx,%rdi  <+143>: callq 0x7c0 <\_ZNSolsEPFRSoS\_E@plt>  <+148>: addl $0x1,-0xc(%rbp)  <+152>: jmp 0x92a <main+32>  <+154>: addl $0x1,-0x8(%rbp)  <+158>: jmpq 0x919 <main+15>  <+163>: mov $0x0,%eax  <+168>: leaveq  <+169>: retq | push rbp to the top of the stack  move rsp into rbp  subtract 0x10 from rsp and store in rsp  move (32 bits) 0x1 into -0x8(%rbp)  compare 0x9 with -0x8(%rbp)  jump to +163 if compare is greater than  move 0x1 into - 0xc(%rbp)  compare 0x9 with -0xc(%rbp)  jump to +154 if compare is greater than  mov 0x8(%rbp) into eax  multiply eax by -0xc(%rbp) and store in eax  move eax into - 0x4(%rbp)  move -0x8(%rbp) into eax  move eax into esi  load rdi into 0x2006da(%rip)  call 0x7e0 <\_ZNSolsEi@plt> function  loads rdi into 0x153(%rip)  move rax into rdi  call 0x7b0 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> function  move rax into rdx  move -0xc(%rbp) into eax  move eax into esi  move rdx into rdi  call 0x7e0 <\_ZNSolsEi@plt> function  load rsi into 0x138(%rip)  move rax into rdi  call x7b0 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> function  move rax into rdx  move -0x4(%rbp) into eax  move eax into esi  move rdx into rdi  call 0x7e0 <\_ZNSolsEi@plt> function  move rax into rdx  move 0x20063d(%rip) into rax  move rax into rsi  move rdx into rdi  call 0x7c0 <\_ZNSolsEPFRSoS\_E@plt> function  add 0x1 to -0xc(%rbp) and store in -0xc(%rbp)  jump to +32  add 0x1 to -0x8(%rbp)  jump to +15  move 0x0 into eax  leave the main function  return |

**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** |
| --- | --- | --- |
| <+0>: push %rbp  <+1>: mov %rsp, %rbp  <+4>: sub $0x10, %rsp | int main() { | Start the main function |
| <+8>: movl $0x1, -0x8(%rbp)  <+15>: cmpl $0x9, -0x8(%rbp)  <+19>: jg 0x9ad <main+163> | for (int i = 1; i <= 9; ++i) { | for 9 loops (increasing i each loop) |
| <+25>: movl $0x1,-0xc(%rbp)  <+32>: cmpl $0x9,-0xc(%rbp)  <+36>: jg 0x9a4 <main+154> | for (int j = 1; j <= 9; ++j) { | for 9 loops (increasing j each loop) |
| <+38>: mov -0x8(%rbp),%eax  <+41>: imul -0xc(%rbp),%eax  <+45>: mov %eax,-0x4(%rbp)  <+48>: mov -0x8(%rbp),%eax  <+51>: mov %eax,%esi  <+53>: lea 0x2006da(%rip),%rdi  # 0x201020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+60>: callq 0x7e0 <\_ZNSolsEi@plt>  <+65>: lea 0x153(%rip),%rsi # 0xaa5  <+72>: mov %rax,%rdi  <+75>: callq 0x7b0 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+80>: mov %rax,%rdx  <+83>: mov -0xc(%rbp),%eax  <+86>: mov %eax,%esi  <+88>: mov %rdx,%rdi  <+91>: callq 0x7e0 <\_ZNSolsEi@plt>  <+96>: lea 0x138(%rip),%rsi # 0xaa9  <+103>: mov %rax,%rdi  <+106>: callq 0x7b0 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+111>: mov %rax,%rdx  <+114>: mov -0x4(%rbp),%eax  <+117>: mov %eax,%esi  <+119>: mov %rdx,%rdi  <+122>: callq 0x7e0 <\_ZNSolsEi@plt>  <+127>: mov %rax,%rdx  <+130>: mov 0x20063d(%rip),%rax # 0x200fd0  <+137>: mov %rax,%rsi  <+140>: mov %rdx,%rdi  <+143>: callq 0x7c0 <\_ZNSolsEPFRSoS\_E@plt> | cout << i << " \* " << j << " = " << i \* j << endl; | print to the consol the current value for i \* j = i\*j  Example:  if i is 3 and j is 2 then the print to the console would be  “3 \* 2 = 6” |
| <+148>: addl $0x1,-0xc(%rbp)  <+152>: jmp 0x92a <main+32> | } | add 1 to j and jump to start of the associated for loop |
| <+154>: addl $0x1,-0x8(%rbp)  <+158>: jmpq 0x919 <main+15> | } | add 1 to i and jump to the start of the associated for loop |
| <+163>: mov $0x0,%eax | return 0; | set 0 as the return value of main |
| <+168>: leaveq  <+169>: retq | } | leave the main function and return the value of main |

## **File Two (assignment4\_2\_fixed)**

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

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| <+0>: push %rbp  <+1>: mov %rsp,%rbp  <+4>: sub $0x30,%rsp  <+8>: mov %fs:0x28,%rax  <+17>: mov %rax,-0x8(%rbp)  <+21>: xor %eax,%eax  <+23>: lea 0x191(%rip),%rsi # 0xba9  <+30>: lea 0x201601(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+37>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+42>: mov %rax,%rdx  <+45>: mov 0x2015a2(%rip),%rax # 0x201fd0  <+52>: mov %rax,%rsi  <+55>: mov %rdx,%rdi  <+58>: callq 0x8a0 <\_ZNSolsEPFRSoS\_E@plt>  <+63>: lea -0x14(%rbp),%rax  <+67>: mov %rax,%rsi  <+70>: lea 0x2016f9(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  <+77>: callq 0x870 <\_ZNSirsERi@plt>  <+82>: mov -0x14(%rbp),%edx  <+85>: mov -0x14(%rbp),%eax  <+88>: imul %eax,%edx  <+91>: mov -0x14(%rbp),%eax  <+94>: imul %edx,%eax  <+97>: mov %eax,-0x14(%rbp)  <+100>: mov -0x14(%rbp),%eax  <+103>: cvtsi2sd %eax,%xmm0  <+107>: movsd 0x15b(%rip),%xmm1 # 0xbc8  <+115>: mulsd %xmm1,%xmm0  <+119>: movsd %xmm0,-0x10(%rbp)  <+124>: lea 0x13a(%rip),%rsi # 0xbb7  <+131>: lea 0x20159c(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+138>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+143>: mov %rax,%rdx  <+146>: mov -0x10(%rbp),%rax  <+150>: mov %rax,-0x28(%rbp)  <+154>: movsd -0x28(%rbp),%xmm0  <+159>: mov %rdx,%rdi  <+162>: callq 0x8d0 <\_ZNSolsEd@plt> <+167>: mov $0x0,%eax  <+172>: mov -0x8(%rbp),%rcx  <+176>: xor %fs:0x28,%rcx  <+185>: je 0xaba <main+192>  <+187>: callq 0x8b0 <\_\_stack\_chk\_fail@plt>  <+192>: leaveq  <+193>: retq | push rbp to the top of the stack  move rsp into rbp  subtract 0x30 from rsp and store in rsp  move fs:0x28 into rax  move rax into -0x8(%rbp)  is eax not equal to eax (sets eax to 0)  loads 0x191(%rip) into rsi  loads 0x201601(%rip) into rdi  call 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> function  move rax into rdx  move 0x2015a2(%rip) into rax  move rax into rsi  move rdx into rdi  call 0x8a0 <\_ZNSolsEPFRSoS\_E@plt> function  load -0x14(%rbp) into rax  move rax into rsi  load 0x2016f9(%rip) into rdi  call the 0x870 <\_ZNSirsERi@plt> function  move -0x14(%rbp) into edx  move -0x14(%rbp) into eax  multiply edx by eax  move -0x14(%rbp) into eax  multiply eax by edx  move eax into -0x14(%rbp)  move -0x14(%rbp) into eax  convert eax into float with double percision  move 0x15b(%rip) to xmm1  multiply xmm0 by xmm1  move xmm0 into -0x10(%rbp)  load 0x13a(%rip) into rsi  load 0x20159c(%rip) into rdi  call 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> function  move rax into rdx  move -0x10(%rbp) into rax  move rax into -0x28(%rbp)  move -0x28(%rbp) into xmm0  move rdx into rdi  call 0x8d0 <\_ZNSolsEd@plt> function  move 0x0 into eax  move -0x8(%rbp) into rcx  check that fs:0x28 is not equal to rcx  jump to +192 if equal  call 0x8b0 <\_\_stack\_chk\_fail@plt> function  leave the main function  return the value of the main function |

**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** |
| --- | --- | --- |
| <+0>: push %rbp  <+1>: mov %rsp,%rbp  <+4>: sub $0x30,%rsp | int main() { | Set up main function memory |
| <+8>: mov %fs:0x28,%rax  <+17>: mov %rax,-0x8(%rbp)  <+21>: xor %eax,%eax  <+23>: lea 0x191(%rip),%rsi # 0xba9  <+30>: lea 0x201601(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4> | float radius = 0;  float volume = 0;  float pi = 3.14; | Create variables for program |
| <+37>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+42>: mov %rax,%rdx  <+45>: mov 0x2015a2(%rip),%rax # 0x201fd0  <+52>: mov %rax,%rsi  <+55>: mov %rdx,%rdi | cout << "Enter Radius:" << endl; | Print message to the console |
| <+58>: callq 0x8a0 <\_ZNSolsEPFRSoS\_E@plt> | cin >> radius; | Get variable from the user for the radius |
| <+63>: lea -0x14(%rbp),%rax  <+67>: mov %rax,%rsi  <+70>: lea 0x2016f9(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  <+77>: callq 0x870 <\_ZNSirsERi@plt>  <+82>: mov -0x14(%rbp),%edx  <+85>: mov -0x14(%rbp),%eax  <+88>: imul %eax,%edx  <+91>: mov -0x14(%rbp),%eax  <+94>: imul %edx,%eax  <+97>: mov %eax,-0x14(%rbp)  <+100>: mov -0x14(%rbp),%eax  <+103>: cvtsi2sd %eax,%xmm0  <+107>: movsd 0x15b(%rip),%xmm1 # 0xbc8  <+115>: mulsd %xmm1,%xmm0  <+119>: movsd %xmm0,-0x10(%rbp)  <+124>: lea 0x13a(%rip),%rsi # 0xbb7 | volume = radius \* radius \* radius \* pi; | get the cubed of radius and multiply that by pi |
| <+131>: lea 0x20159c(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+138>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+143>: mov %rax,%rdx  <+146>: mov -0x10(%rbp),%rax  <+150>: mov %rax,-0x28(%rbp)  <+154>: movsd -0x28(%rbp),%xmm0  <+159>: mov %rdx,%rdi  <+162>: callq 0x8d0 <\_ZNSolsEd@plt> <+167>: mov $0x0,%eax  <+172>: mov -0x8(%rbp),%rcx  <+176>: xor %fs:0x28,%rcx  <+185>: je 0xaba <main+192>  <+187>: callq 0x8b0 <\_\_stack\_chk\_fail@plt> | cout << "The volume is: " << fixed << setprecision(2) << volume; | Print the volume to the console with the volume set to precision 2 (2 places past the decimal point) |
| <+192>: leaveq | return 0; | set the value of main to 0 |
| <+193>: retq | } | return the value of main |

## **File Three (assignment4\_3\_fixed)**

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

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| <+0>: push %rbp  <+1>: mov %rsp,%rbp  <+4>: sub $0x20,%rsp  <+8>: mov %fs:0x28,%rax  <+17>: mov %rax,-0x8(%rbp)  <+21>: xor %eax,%eax  <+23>: movl $0x1,-0xc(%rbp)  <+30>: lea 0x256(%rip),%rsi # 0xc35  <+37>: lea 0x20163a(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+44>: callq 0x860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+49>: mov %rax,%rdx  <+52>: mov 0x2015db(%rip),%rax # 0x201fd0  <+59>: mov %rax,%rsi  <+62>: mov %rdx,%rdi  <+65>: callq 0x870 <\_ZNSolsEPFRSoS\_E@plt>  <+70>: lea -0x18(%rbp),%rax  <+74>: mov %rax,%rsi  <+77>: lea 0x201732(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  <+84>: callq 0x840 <\_ZNSirsERi@plt>  <+89>: mov -0x18(%rbp),%eax  <+92>: sub $0x1,%eax  <+95>: mov %eax,-0xc(%rbp)  <+98>: movl $0x1,-0x10(%rbp)  <+105>: mov -0x18(%rbp),%eax  <+108>: cmp %eax,-0x10(%rbp)  <+111>: jg 0xa9d <main+227>  <+113>: movl $0x1,-0x14(%rbp)  <+120>: mov -0x14(%rbp),%eax  <+123>: cmp -0xc(%rbp),%eax  <+126>: jg 0xa53 <main+153>  <+128>: lea 0x209(%rip),%rsi # 0xc4a  <+135>: lea 0x2015d8(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+142>: callq 0x860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+147>: addl $0x1,-0x14(%rbp)  <+151>: jmp 0xa32 <main+120>  <+153>: subl $0x1,-0xc(%rbp)  <+157>: movl $0x1,-0x14(%rbp)  <+164>: mov -0x10(%rbp),%eax  <+167>: add %eax,%eax  <+169>: sub $0x1,%eax  <+172>: cmp %eax,-0x14(%rbp)  <+175>: jg 0xa84 <main+202>  <+177>: lea 0x1da(%rip),%rsi # 0xc4c  <+184>: lea 0x2015a7(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+191>: callq 0x860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+196>: addl $0x1,-0x14(%rbp)  <+200>: jmp 0xa5e <main+164>  <+202>: lea 0x1c3(%rip),%rsi # 0xc4e  <+209>: lea 0x20158e(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+216>: callq 0x860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+221>: addl $0x1,-0x10(%rbp)  <+225>: jmp 0xa23 <main+105>  <+227>: movl $0x1,-0xc(%rbp)  <+234>: movl $0x1,-0x10(%rbp)  <+241>: mov -0x18(%rbp),%eax  <+244>: sub $0x1,%eax  <+247>: cmp %eax,-0x10(%rbp)  <+250>: jg 0xb2b <main+369>  <+252>: movl $0x1,-0x14(%rbp)  <+259>: mov -0x14(%rbp),%eax  <+262>: cmp -0xc(%rbp),%eax  <+265>: jg 0xade <main+292>  <+267>: lea 0x17e(%rip),%rsi # 0xc4a  <+274>: lea 0x20154d(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+281>: callq 0x860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+286>: addl $0x1,-0x14(%rbp)  <+290>: jmp 0xabd <main+259>  <+292>: addl $0x1,-0xc(%rbp)  <+296>: movl $0x1,-0x14(%rbp)  <+303>: mov -0x18(%rbp),%eax  <+306>: sub -0x10(%rbp),%eax  <+309>: add %eax,%eax  <+311>: sub $0x1,%eax  <+314>: cmp %eax,-0x14(%rbp)  <+317>: jg 0xb12 <main+344>  <+319>: lea 0x14c(%rip),%rsi # 0xc4c  <+326>: lea 0x201519(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+333>: callq 0x860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+338>: addl $0x1,-0x14(%rbp)  <+342>: jmp 0xae9 <main+303>  <+344>: lea 0x135(%rip),%rsi # 0xc4e  <+351>: lea 0x201500(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+358>: callq 0x860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+363>: addl $0x1,-0x10(%rbp)  <+367>: jmp 0xaab <main+241>  <+369>: mov $0x1,%eax  <+374>: mov -0x8(%rbp),%rcx  <+378>: xor %fs:0x28,%rcx  <+387>: je 0xb44 <main+394>  <+389>: callq 0x880 <\_\_stack\_chk\_fail@plt>  <+394>: leaveq  <+395>: retq | push rbp to the top of the stack  move rsp into rbp  subtract 0x20 from rsp  move fs:0x28 into rax  move rax into -0x8(%rbp)  compare to see if eax is not equal to eax  move 0x1 to -0xc(%rbp)  load 0x256(%rip) into rdi  load 0x20163a(%rip) into rdi  call 0x860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> function  move rax into rdx  move 0x2015db(%rip) into rax  move rax into rsi  move rdx into rdi  call 0x870 <\_ZNSolsEPFRSoS\_E@plt> function  load 0x18(%rbp) into rax  move rax into rsi  load 0x201732(%rip) into rdi  call 0x840 <\_ZNSirsERi@plt> function  move -0x18(%rbp) into eax  subtract 0x1 from eax  move eax into -0xc(%rbp)  move 0x1 into -0x10(%rbp)  move -0x18(%rbp) into eax  compare eax with - 0x10(%rbp)  jump to +227 if greater than  move 0x1 into - 0x14(%rbp)  move -0x14(%rbp) into eax  compare -0xc(%rbp) with eax  jump to +153 if greater than  load 0x209(%rip) into rsi  load 0x2015d8(%rip) into rdi  call 0x860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> function  add 0x1 into -0x14(%rbp)  jump to +120  subtract 0x1 from -0xc(%rbp)  move 0x1 into -0x14(%rbp)  move -0x10(%rbp) into eax  add eax to eax  subtract 0x1 from eax  compare eax with -0x14(%rbp)  jump to +202 if greater than  load 0x1da(%rip) into rsi  load 0x2015a7(%rip) into rdi  call 0x860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> function  add 0x1 to -0x14(%rbp)  jump to +164  lea 0x1c3(%rip) to rsi  load 0x20158e(%rip) to rdi  call 0x860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> function  add 0x1 to -0x10(%rbp)  jump to +105  move 0x1 into -0xc(%rbp)  move 0x1 into -0x10(%rbp)  move -0x18(%rbp) into eax  subtract 0x1 eax  compare eax to -0x10(%rbp)  jump to +369 if greater than  move 0x1 into -0x14(%rbp)  move -0x14(%rbp) into eax  compare -0xc(%rbp) into eax  jump to +292 if greater than  load 0x17e(%rip) into rsi  load 0x20154d(%rip) into rdi  call 0x860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> function  add 0x1 to -0x14(%rbp)  jump to +259  add 0x1 to -0xc(%rbp)  move -0x18(%rbp) to eax  move -0x18(%rbp) to eax  subtract -0x10(%rbp) to eax  add eax to eax  subtract 0x1 from eax  compare eax to -0x14(%rbp)  jump to +344 if grater than  load 0x14c(%rip) into rsi  load 0x201519(%rip) into rdi  call callq 0x860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> function  add 0x1 to -0x14(%rbp)  jump to +303  load 0x135(%rip) to rsi  load 0x201500(%rip) to rdi  call 0x860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  add 0x1 to -0x10(%rbp)  jump to +241  move 0x1 to eax  move -0x8(%rbp) to rcx  check if %fs:0x28 does not equal  jump to +394 if equal %rcx  call callq 0x880 <\_\_stack\_chk\_fail@plt>  leave the main function  return value of main function |

**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** |
| --- | --- | --- |
| <+0>: push %rbp  <+1>: mov %rsp,%rbp  <+4>: sub $0x20,%rsp  <+8>: mov %fs:0x28,%rax | int numRows = 0; | Start of main function, initialization of int variable for storing user inputted number of rows (half of the diamond) |
| <+17>: mov %rax,-0x8(%rbp)  <+21>: xor %eax,%eax  <+23>: movl $0x1,-0xc(%rbp) | char rowChar = '\*'; | Variable for storing the character that will be printed for the diamond |
| <+30>: lea 0x256(%rip),%rsi # 0xc35  <+37>: lea 0x20163a(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+44>: callq 0x860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> | cout << "Enter number of rows" << endl; | Print to the console to ask user for number of rows |
| <+49>: mov %rax,%rdx  <+52>: mov 0x2015db(%rip),%rax # 0x201fd0  <+59>: mov %rax,%rsi  <+62>: mov %rdx,%rdi  <+65>: callq 0x870 <\_ZNSolsEPFRSoS\_E@plt>  <+70>: lea -0x18(%rbp),%rax  <+74>: mov %rax,%rsi  <+77>: lea 0x201732(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  <+84>: callq 0x840 <\_ZNSirsERi@plt> | cin >> numRows; | recieve number of rows from the user |
| <+89>: mov -0x18(%rbp),%eax  <+92>: sub $0x1,%eax  <+95>: mov %eax,-0xc(%rbp)  <+98>: movl $0x1,-0x10(%rbp)  <+105>: mov -0x18(%rbp),%eax  <+108>: cmp %eax,-0x10(%rbp)  <+111>: jg 0xa9d <main+227> | // loop for upper half of diamond  for (int i = 0; i <= numRows; ++i) { | first loop for iterating through the top half of the diamond |
| <+113>: movl $0x1,-0x14(%rbp)  <+120>: mov -0x14(%rbp),%eax  <+123>: cmp -0xc(%rbp),%eax  <+126>: jg 0xa53 <main+153> | // loop for spaces before character(s)  for (int j = 1; j <= numRows - i; ++j) { | second loop for printing the spaces before the characters for that line of the diamond (no need to code for the second half of the spaces since it is all whitespace already) |
| <+128>: lea 0x209(%rip),%rsi # 0xc4a  <+135>: lea 0x2015d8(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+142>: callq 0x860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> | cout << " "; | actual printing of the space before the characters |
| <+147>: addl $0x1,-0x14(%rbp)  <+151>: jmp 0xa32 <main+120>  <+153>: subl $0x1,-0xc(%rbp)  <+157>: movl $0x1,-0x14(%rbp)  <+164>: mov -0x10(%rbp),%eax  <+167>: add %eax,%eax  <+169>: sub $0x1,%eax | } | end of second loop |
| <+172>: cmp %eax,-0x14(%rbp)  <+175>: jg 0xa84 <main+202> | // loop to print character(s)  for (int j = 1; j <= (2 \* i) - 1; ++j) { | third loop, this one is for adding the correct number of characters for the current row  (2 \* row number – 1) |
| <+177>: lea 0x1da(%rip),%rsi # 0xc4c  <+184>: lea 0x2015a7(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+191>: callq 0x860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> | cout << rowChar; | print the coded symbol to the console |
| <+196>: addl $0x1,-0x14(%rbp)  <+200>: jmp 0xa5e <main+164> | } | end of the thrid loop |
| <+202>: lea 0x1c3(%rip),%rsi # 0xc4e  <+209>: lea 0x20158e(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+216>: callq 0x860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> | cout << endl; | go to the next line once all the symbols are added |
| <+221>: addl $0x1,-0x10(%rbp)  <+225>: jmp 0xa23 <main+105> | } | end of the first loop, meaning the top half of the diamond is complete |
| <+227>: movl $0x1,-0xc(%rbp)  <+234>: movl $0x1,-0x10(%rbp)  <+241>: mov -0x18(%rbp),%eax  <+244>: sub $0x1,%eax  <+247>: cmp %eax,-0x10(%rbp)  <+250>: jg 0xb2b <main+369> | // loop for lower hald of diamond  for (int i = numRows - 1; 1 <= i; --i) { | first loop for the second half of the loop, this time i needs to be set to the last iteration of the top half of the diamond, but will be counting down each loop instead of counting up |
| <+252>: movl $0x1,-0x14(%rbp)  <+259>: mov -0x14(%rbp),%eax  <+262>: cmp -0xc(%rbp),%eax  <+265>: jg 0xade <main+292> | // loop to print spaces before the character(s)  for (int j = 1; j <= numRows - i; ++j) { | second loop of part 2, for adding the correct number of spaces before the characters |
| <+267>: lea 0x17e(%rip),%rsi # 0xc4a  <+274>: lea 0x20154d(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+281>: callq 0x860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> | cout << " "; | printing the spaces before the characters |
| <+286>: addl $0x1,-0x14(%rbp)  <+290>: jmp 0xabd <main+259> | } | end of the second loop |
| <+292>: addl $0x1,-0xc(%rbp)  <+296>: movl $0x1,-0x14(%rbp)  <+303>: mov -0x18(%rbp),%eax  <+306>: sub -0x10(%rbp),%eax  <+309>: add %eax,%eax  <+311>: sub $0x1,%eax  <+314>: cmp %eax,-0x14(%rbp)  <+317>: jg 0xb12 <main+344> | // loop to print the character(s)  for (int j = 1; j <= (2 \* i) - 1; ++j) { | third loop for printing the symbols to the console |
| <+319>: lea 0x14c(%rip),%rsi # 0xc4c  <+326>: lea 0x201519(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+333>: callq 0x860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> | cout << rowChar; | printing the symbol to the console |
| <+338>: addl $0x1,-0x14(%rbp)  <+342>: jmp 0xae9 <main+303> | } | end of the third loop |
| <+344>: lea 0x135(%rip),%rsi # 0xc4e  <+351>: lea 0x201500(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+358>: callq 0x860 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> | cout << endl; | move to the next line once all the characters have been added |
| <+363>: addl $0x1,-0x10(%rbp)  <+367>: jmp 0xaab <main+241> | } | end of the second half of the diamond |
| <+369>: mov $0x1,%eax  <+374>: mov -0x8(%rbp),%rcx  <+378>: xor %fs:0x28,%rcx  <+387>: je 0xb44 <main+394>  <+389>: callq 0x880 <\_\_stack\_chk\_fail@plt>  <+394>: leaveq  <+395>: retq | } | end of the main function |

## **File Four (assignment4\_4\_fixed)**

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

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| <+0>: push %rbp  <+1>: mov %rsp,%rbp  <+4>: sub $0x30, %rsp  <+8>: mov %fs:0x28,%rax  <+17>: mov %rax,-0x8(%rbp)  <+21>: xor %eax,%eax  <+23>: movq $0x0,-0x20(%rbp)  <+31>: movq $0x1,-0x18(%rbp)  <+39>: lea 0x201(%rip),%rsi # 0xc29  <+46>: lea 0x2015f1(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+53>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+58>: mov %rax,%rdx  <+61>: mov 0x201592(%rip),%rax # 0x201fd0  <+68>: mov %rax,%rsi  <+71>: mov %rdx,%rdi  <+74>: callq 0x8a0 <\_ZNSolsEPFRSoS\_E@plt>  <+79>: lea -0x28(%rbp),%rax  <+83>: mov %rax,%rsi  <+86>: lea 0x2016e9(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  <+93>: callq 0x870 <\_ZNSirsERl@plt>  <+98>: mov -0x28(%rbp),%rax  <+102>: test %rax,%rax  <+105>: je 0xaec <main+242>  <+111>: mov -0x28(%rbp),%rcx  <+115>: movabs $0x6666666666666667,%rdx  <+125>: mov %rcx,%rax  <+128>: imul %rdx  <+131>: sar $0x2,%rdx  <+135>: mov %rcx,%rax  <+138>: sar $0x3f,%rax  <+142>: sub %rax,%rdx  <+145>: mov %rdx,%rax  <+148>: mov %rax,-0x10(%rbp)  <+152>: mov -0x10(%rbp),%rdx  <+156>: mov %rdx,%rax  <+159>: shl $0x2,%rax  <+163>: add %rdx,%rax  <+166>: add %rax,%rax  <+169>: sub %rax,%rcx  <+172>: mov %rcx,%rax  <+175>: mov %rax,-0x10(%rbp)  <+179>: mov -0x10(%rbp),%rax  <+183>: imul -0x18(%rbp),%rax  <+188>: add %rax,-0x20(%rbp)  <+192>: shlq -0x18(%rbp)  <+196>: mov -0x28(%rbp),%rcx  <+200>: movabs $0x6666666666666667,%rdx  <+210>: mov %rcx,%rax  <+213>: imul %rdx  <+216>: sar $0x2,%rdx  <+220>: mov %rcx,%rax  <+223>: sar $0x3f,%rax  <+227>: sub %rax,%rdx  <+230>: mov %rdx,%rax  <+233>: mov %rax,-0x28(%rbp)  <+237>: jmpq 0xa5c <main+98>  <+242>: lea 0x155(%rip),%rsi # 0xc48  <+249>: lea 0x201526(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+256>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+261>: mov %rax,%rdx  <+264>: mov -0x20(%rbp),%rax  <+268>: mov %rax,%rsi  <+271>: mov %rdx,%rdi  <+274>: callq 0x8d0 <\_ZNSolsEl@plt>  <+279>: mov %rax,%rdx  <+282>: mov 0x2014b5(%rip),%rax # 0x201fd0  <+289>: mov %rax,%rsi  <+292>: mov %rdx,%rdi  <+295>: callq 0x8a0 <\_ZNSolsEPFRSoS\_E@plt>  <+300>: mov $0x0,%eax  <+305>: mov -0x8(%rbp),%rsi  <+309>: xor %fs:0x28,%rsi  <+318>: je 0xb3f <main+325>  <+320>: callq 0x8b0 <\_\_stack\_chk\_fail@plt>  <+325>: leaveq  <+326>: retq | push rbp to the top of the stack  move rsp into rbp  subtract $0x30, from rsp  move %fs:0x28 into rax  move rax into -0x8(%rbp)  check if eax is not equal to eax  move 0x0 into -0x20(%rbp)  move 0x1 into -0x18(%rbp)  load 0x201(%rip) into rsi  load 0x2015f1(%rip) into rdi  call function  move rax into rdx  move 0x201592(%rip) into rax  move rax into rsi  move rdx into rdi  call function  load -0x28(%rbp) into rax  move rax into rsi  load 0x2016e9(%rip) into rdi  call function  move -0x28(%rbp) into rax  set zero flag if rax is 0  jump to +242 if equals  move -0x28(%rbp) to rcx  move(64 bit) 0x6666666666666667 into rdx  move rcx into rax  multiply rdx by itself  Shift arithmic right rdx by 0x2  move rcx into rax  shift arithmic right rax by 0x3f  subtract rax from rdx  move rdx to rax  move rax into -0x10(%rbp)  move -0x10(%rbp) into rdx  move rdx into rax  shift left rax by 0x2  add rdx to rax  add rax to rax  subtract rax from rcx  move rcx to rax  move rax into -0x10(%rbp)  move -0x10(%rbp) into rax  multiply -0x18(%rbp) by rax and store in rax  add rax to 0x20  shift left 0x18(%rbp) by itself  move -0x28(%rbp) to rcx  move 0x6666666666666667 to rdx  move rcx to rax  multiply rdx by itself  shift right rdx by 0x2  move rcx to rax  shift right rax by 0x3f  subtract rax from rdx  move rdx to rax  move rax to -0x28(%rbp)  jump to +98  load 0x155(%rip) into rsi  load 0x201526(%rip) into rdi  call function  move rax into rdx  move mov -0x20(%rbp) to rax  move rax to rsi  move rdx to rdi  call function  move rax to rdx  move 0x2014b5(%rip) to rax  move rax to rsi  move rdx to rdi  call function  move 0x0 to eax  move -0x8(%rbp) to rsi  check if %fs:0x28 is not equal to rsi  jump to 352 if equal  call function  leave main function  return value of main function |

**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** |
| --- | --- | --- |
| <+0>: push %rbp  <+1>: mov %rsp,%rbp  <+4>: sub $0x30, %rsp | int main() { | start of the main function |
| <+8>: mov %fs:0x28,%rax  <+17>: mov %rax,-0x8(%rbp)  <+21>: xor %eax,%eax  <+23>: movq $0x0,-0x20(%rbp)  <+31>: movq $0x1,-0x18(%rbp) | int binNum, hexDecNum = 0, base = 1; | initialize variables for the binary number, the hexadecimal number, and the base value (2^0 =1) |
| <+39>: lea 0x201(%rip),%rsi # 0xc29  <+46>: lea 0x2015f1(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+53>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> | cout << "Enter the binary number:" << endl; | print to the console asking user for the binary number they wish to convert |
| <+58>: mov %rax,%rdx  <+61>: mov 0x201592(%rip),%rax # 0x201fd0  <+68>: mov %rax,%rsi  <+71>: mov %rdx,%rdi  <+74>: callq 0x8a0 <\_ZNSolsEPFRSoS\_E@plt>  <+79>: lea -0x28(%rbp),%rax  <+83>: mov %rax,%rsi  <+86>: lea 0x2016e9(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  <+93>: callq 0x870 <\_ZNSirsERl@plt> | cin >> binNum; | receive the binary number from the user |
| <+98>: mov -0x28(%rbp),%rax  <+102>: test %rax,%rax  <+105>: je 0xaec <main+242> | while (binNum) { | while binNum is not 0 |
| <+111>: mov -0x28(%rbp),%rcx  <+115>: movabs $0x6666666666666667,%rdx  <+125>: mov %rcx,%rax  <+128>: imul %rdx  <+131>: sar $0x2,%rdx | int lastDigit = binNum % 10; | the last digit to be converted is the remainder of the binary number divided by 10 |
| <+135>: mov %rcx,%rax  <+138>: sar $0x3f,%rax  <+142>: sub %rax,%rdx  <+145>: mov %rdx,%rax  <+148>: mov %rax,-0x10(%rbp)  <+152>: mov -0x10(%rbp),%rdx  <+156>: mov %rdx,%rax  <+159>: shl $0x2,%rax | binNum /= 10; | the binary number is then divided by 10 (not containing any remainder) |
| <+163>: add %rdx,%rax  <+166>: add %rax,%rax  <+169>: sub %rax,%rcx  <+172>: mov %rcx,%rax  <+175>: mov %rax,-0x10(%rbp)  <+179>: mov -0x10(%rbp),%rax  <+183>: imul -0x18(%rbp),%rax  <+188>: add %rax,-0x20(%rbp)  <+192>: shlq -0x18(%rbp)  <+196>: mov -0x28(%rbp),%rcx  <+200>: movabs $0x6666666666666667,%rdx | hexDecNum += lastDigit \* base; | the last digit multiplied by the base is added to the hexadecimal number |
| <+210>: mov %rcx,%rax  <+213>: imul %rdx  <+216>: sar $0x2,%rdx  <+220>: mov %rcx,%rax  <+223>: sar $0x3f,%rax  <+227>: sub %rax,%rdx  <+230>: mov %rdx,%rax  <+233>: mov %rax,-0x28(%rbp) | base \*= 2; | the base is then multiplied by 2 |
| <+237>: jmpq 0xa5c <main+98> | } | end of the while loop |
| <+242>: lea 0x155(%rip),%rsi # 0xc48  <+249>: lea 0x201526(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+256>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> | cout << endl << "Equivalent hexadecimal value: " << hexDecNum; | print to the console the converted number |
| <+261>: mov %rax,%rdx  <+264>: mov -0x20(%rbp),%rax  <+268>: mov %rax,%rsi  <+271>: mov %rdx,%rdi  <+274>: callq 0x8d0 <\_ZNSolsEl@plt>  <+279>: mov %rax,%rdx  <+282>: mov 0x2014b5(%rip),%rax # 0x201fd0  <+289>: mov %rax,%rsi  <+292>: mov %rdx,%rdi  <+295>: callq 0x8a0 <\_ZNSolsEPFRSoS\_E@plt>  <+300>: mov $0x0,%eax  <+305>: mov -0x8(%rbp),%rsi  <+309>: xor %fs:0x28,%rsi  <+318>: je 0xb3f <main+325>  <+320>: callq 0x8b0 <\_\_stack\_chk\_fail@plt>  <+325>: leaveq | return 0; | set the value of main to 0 |
| <+326>: retq | } | end of main function, return the value of the main function |