CS 410 Binary to C++ Activity

File One: assignment4_1.o	
Blocks of Assembly Code	**Explanation of Functionality**

movl $0x1,-0x8(%rbp)
cmpl $0x9,-0x8(%rbp)
jg 0xa3 <main+163>
	Assign 1 to -0x8(%rbp)
	Compare -0x8(%rbp) to 9
	If true, jump to 0xa3 <main+163>

movl $0x1,-0xc(%rbp)
cmpl $0x9,-0xc(%rbp)
jg 0x9a <main+154>
	Assign 1 to -0xc(%rbp)
	Compare -0xc(%rbp) to 9
	If true, jump to 0xa3 <main+154>

mov -0x8(%rbp),%eax
imul -0xc(%rbp),%eax
mov %eax,-0x4(%rbp)
	Move -0x8(%rbp) to %eax
	Multiply -0xc(%rbp) by %eax
	Assign the result to -0x4(%rbp)

mov -0x8(%rbp),%eax
mov %eax,%esi
lea 0x0(%rip),%rdi
callq 0x41 <main+65>
lea 0x0(%rip),%rsi
mov %rax,%rdi
callq 0x50 <main+80>
mov %rax,%rdx
mov -0xc(%rbp),%eax
mov %eax,%esi
mov %rdx,%rdi
callq 0x60 <main+96>
lea 0x0(%rip),%rsi
mov %rax,%rdi
callq 0x6f <main+111>
mov %rax,%rdx
mov -0x4(%rbp),%eax
mov %eax,%esi
mov %rdx,%rdi
callq 0x7f <main+127>
mov %rax,%rdx
mov 0x0(%rip),%rax
mov %rax,%rsi
mov %rdx,%rdi
callq 0x94 <main+148>
	Print output

addl $0x1,-0xc(%rbp) jmp 0x20 <main+32>	Add 1 to -0xc(%rbp) Jump back to 0x20 <main+32>
addl $0x1,-0x8(%rbp) jmpq 0xf <main+15>	Add 1 to -0x8(%rbp) Jump back to 0xf <main+15>
mov $0x0,%eax leaveq retq	Exit main method and return zero

File One: assignment4_1.o		
Blocks of Assembly Code	**C++ Code**	**Explanation of Functionality**

movl $0x1,-0x8(%rbp)
cmpl $0x9,-0x8(%rbp)
jg 0xa3 <main+163>
 	
addl $0x1,-0x8(%rbp)
jmpq 0xf <main+15>
	for (int x = 1; x <= 9; x++)	Create a loop that iterates 9 times

movl $0x1,-0xc(%rbp)
cmpl $0x9,-0xc(%rbp)
jg 0x9a <main+154>
	for (int y = 1; y <= 9; y++)	Create a loop that iterates 9 times

mov -0x8(%rbp),%eax
imul -0xc(%rbp),%eax
mov %eax,-0x4(%rbp)
	answer = x * y;	Multiply two values to get the answer

mov -0x8(%rbp),%eax
mov %eax,%esi
lea 0x0(%rip),%rdi
callq 0x41 <main+65>
lea 0x0(%rip),%rsi
mov %rax,%rdi
callq 0x50 <main+80>
mov %rax,%rdx
mov -0xc(%rbp),%eax
mov %eax,%esi
mov %rdx,%rdi
callq 0x60 <main+96>
lea 0x0(%rip),%rsi
mov %rax,%rdi
callq 0x6f <main+111>
mov %rax,%rdx
mov -0x4(%rbp),%eax
mov %eax,%esi
mov %rdx,%rdi
callq 0x7f <main+127>
mov %rax,%rdx
mov 0x0(%rip),%rax
mov %rax,%rsi
mov %rdx,%rdi
callq 0x94 <main+148>
	cout << x << " * " << y << " = " << answer << endl;	Print output

mov $0x0,%eax leaveq retq	return 0;	Return 0 and exit the program

File Two: assignment4_2.o	
Blocks of Assembly Code	**Explanation of Functionality**

sub $0x30,%rsp
mov %fs:0x28,%rax
mov %rax,-0x8(%rbp)
xor %eax,%eax
lea 0x0(%rip),%rsi
lea 0x0(%rip),%rdi
callq 0x2a <main+42>
mov %rax,%rdx
mov 0x0(%rip),%rax
mov %rax,%rsi
mov %rdx,%rdi	

callq 0x3f <main+63>
lea -0x14(%rbp),%rax
mov %rax,%rsi
lea 0x0(%rip),%rdi
callq 0x52 <main+82>
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 0x0(%rip),%xmm1
mulsd %xmm1,%xmm0
movsd %xmm0,-0x10(%rbp)
lea 0x0(%rip),%rsi
lea 0x0(%rip),%rdi
callq 0x8f <main+143>	

mov %rax,%rdx
mov -0x10(%rbp),%rax
mov %rax,-0x28(%rbp)
movsd -0x28(%rbp),%xmm0
mov %rdx,%rdi
callq 0xa7 <main+167>
mov $0x0,%eax
mov -0x8(%rbp),%rcx
xor %fs:0x28,%rcx	

je 0xc0 <main+192> callq 0xc0 <main+192> leaveq retq	

File Two: assignment4_2.o		
Blocks of Assembly Code	**C++ Code**	**Explanation of Functionality**

sub $0x30,%rsp
mov %fs:0x28,%rax
mov %rax,-0x8(%rbp)
xor %eax,%eax
lea 0x0(%rip),%rsi
lea 0x0(%rip),%rdi
callq 0x2a <main+42>
mov %rax,%rdx
mov 0x0(%rip),%rax
mov %rax,%rsi
mov %rdx,%rdi
	float vol, pi = 3.14159;
	int rad;
	
	cout << "Enter Radius: " << endl;
	cin >> rad;
		Set value for pi
		Print “Enter Radius: ”
		Read in and assign user input

callq 0x3f <main+63>
lea -0x14(%rbp),%rax
mov %rax,%rsi
lea 0x0(%rip),%rdi
callq 0x52 <main+82>
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 0x0(%rip),%xmm1
mulsd %xmm1,%xmm0
movsd %xmm0,-0x10(%rbp)
lea 0x0(%rip),%rsi
lea 0x0(%rip),%rdi
callq 0x8f <main+143>
	vol = rad * rad * rad * pi;	Multiply values to get the volume

mov %rax,%rdx
mov -0x10(%rbp),%rax
mov %rax,-0x28(%rbp)
movsd -0x28(%rbp),%xmm0
mov %rdx,%rdi
callq 0xa7 <main+167>
mov $0x0,%eax
mov -0x8(%rbp),%rcx
xor %fs:0x28,%rcx
	cout << "The volume is: " << vol << endl;	Print “The volume is: “ followed by the calculated volume

je 0xc0 <main+192> callq 0xc0 <main+192> leaveq retq	return 0;	Exit the main method and return zero

File Three: assignment4_3.o	
Blocks of Assembly Code	**Explanation of Functionality**
push %rbp	
mov %rsp,%rbp	
sub $0x20,%rsp	
mov %fs:0x28,%rax	

```
mov    %rax,-0x8(%rbp)
xor    %eax,%eax
movl   $0x1,-0xc(%rbp)
lea    0x0(%rip),%rsi
lea    0x0(%rip),%rdi
callq  0x31 <main+49>
mov    %rax,%rdx
mov    0x0(%rip),%rax
mov    %rax,%rsi
mov    %rdx,%rdi
callq  0x46 <main+70>
lea    -0x18(%rbp),%rax
mov    %rax,%rsi
lea    0x0(%rip),%rdi
callq  0x59 <main+89>
mov    -0x18(%rbp),%eax
sub    $0x1,%eax
mov    %eax,-0xc(%rbp)
movl   $0x1,-0x10(%rbp)
mov    -0x18(%rbp),%eax
cmp    %eax,-0x10(%rbp)
jg     0xe3 <main+227>
movl   $0x1,-0x14(%rbp)
mov    -0x14(%rbp),%eax
cmp    -0xc(%rbp),%eax
jg     0x99 <main+153>
lea    0x0(%rip),%rsi
lea    0x0(%rip),%rdi
callq  0x93 <main+147>
addl   $0x1,-0x14(%rbp)
jmp    0x78 <main+120>
subl   $0x1,-0xc(%rbp)
movl   $0x1,-0x14(%rbp)
mov    -0x10(%rbp),%eax
add    %eax,%eax
sub    $0x1,%eax
cmp    %eax,-0x14(%rbp)
jg     0xca <main+202>
lea    0x0(%rip),%rsi
lea    0x0(%rip),%rdi
callq  0xc4 <main+196>
addl   $0x1,-0x14(%rbp)
jmp    0xa4 <main+164>
lea    0x0(%rip),%rsi
lea    0x0(%rip),%rdi
callq  0xdd <main+221>
addl   $0x1,-0x10(%rbp)
jmp    0x69 <main+105>
movl   $0x1,-0xc(%rbp)
movl   $0x1,-0x10(%rbp)
```

mov -0x18(%rbp),%eax
sub $0x1,%eax
cmp %eax,-0x10(%rbp)
jg 0x171 <main+369>
movl $0x1,-0x14(%rbp)
mov -0x14(%rbp),%eax
cmp -0xc(%rbp),%eax
jg 0x124 <main+292>
lea 0x0(%rip),%rsi
lea 0x0(%rip),%rdi
callq 0x11e <main+286>
addl $0x1,-0x14(%rbp)
jmp 0x103 <main+259>
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 0x158 <main+344>
lea 0x0(%rip),%rsi
lea 0x0(%rip),%rdi
callq 0x152 <main+338>
addl $0x1,-0x14(%rbp)
jmp 0x12f <main+303>
lea 0x0(%rip),%rsi
lea 0x0(%rip),%rdi
callq 0x16b <main+363>
addl $0x1,-0x10(%rbp)
jmp 0xf1 <main+241>
mov $0x1,%eax
mov -0x8(%rbp),%rcx
xor %fs:0x28,%rcx
je 0x18a <main+394>
callq 0x18a <main+394>
leaveq
retq	

File Three: assignment4_3.o		
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 0x0(%rip),%rsi
lea 0x0(%rip),%rdi
callq 0x31 <main+49>
mov %rax,%rdx
mov 0x0(%rip),%rax
mov %rax,%rsi
mov %rdx,%rdi
callq 0x46 <main+70>
lea -0x18(%rbp),%rax
mov %rax,%rsi
lea 0x0(%rip),%rdi
callq 0x59 <main+89>
mov -0x18(%rbp),%eax
sub $0x1,%eax
mov %eax,-0xc(%rbp)
	int x, y, rows, space;
	cout << " Enter number of rows" << endl;
	cin >> rows;
	z = rows - 1;
		Print “Enter number of rows.”
		Read in and assign user input
		Subtract 1 from rows and assign to

movl $0x1,-0x10(%rbp)
mov -0x18(%rbp),%eax
cmp %eax,-0x10(%rbp)
jg 0xe3 <main+227>
	for (x = 1; x <= rows; x++)	

movl $0x1,-0x14(%rbp)
mov -0x14(%rbp),%eax
cmp -0xc(%rbp),%eax
jg 0x99 <main+153>
	for (y = 1; y <= z; y++)	

File Four: assignment4_4.o	
Blocks of Assembly Code	**Explanation of Functionality**
push %rbp	
mov %rsp,%rbp	
sub $0x30,%rsp	
mov %fs:0x28,%rax	

```
mov    %rax,-0x8(%rbp)
xor    %eax,%eax
movq   $0x0,-0x20(%rbp)
movq   $0x1,-0x18(%rbp)
lea    0x0(%rip),%rsi
lea    0x0(%rip),%rdi
callq  0x3a <main+58>
mov    %rax,%rdx
mov    0x0(%rip),%rax
mov    %rax,%rsi
mov    %rdx,%rdi
callq  0x4f <main+79>
lea    -0x28(%rbp),%rax
mov    %rax,%rsi
lea    0x0(%rip),%rdi
callq  0x62 <main+98>
mov    -0x28(%rbp),%rax
test   %rax,%rax
je     0xf2 <main+242>
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,-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 0x62 <main+98>
lea 0x0(%rip),%rsi # 0xf9 <main+249>
lea 0x0(%rip),%rdi # 0x100 <main+256>
callq 0x105 <main+261>
mov %rax,%rdx
mov -0x20(%rbp),%rax
mov %rax,%rsi
mov %rdx,%rdi
callq 0x117 <main+279>
mov %rax,%rdx
mov 0x0(%rip),%rax # 0x121 <main+289>
mov %rax,%rsi
mov %rdx,%rdi
callq 0x12c <main+300>
mov $0x0,%eax
mov -0x8(%rbp),%rsi
xor %fs:0x28,%rsi
je 0x145 <main+325>
callq 0x145 <main+325>
leaveq
retq	

File Four: assignment4_4.o		
Blocks of Assembly Code	**C++ Code**	**Explanation of Functionality**

push %rbp
mov %rsp,%rbp
sub $0x30,%rsp
mov %fs:0x28,%rax		

mov %rax,-0x8(%rbp)
xor %eax,%eax
movq $0x0,-0x20(%rbp)
movq $0x1,-0x18(%rbp)
lea 0x0(%rip),%rsi
lea 0x0(%rip),%rdi
callq 0x3a <main+58>
mov %rax,%rdx
mov 0x0(%rip),%rax
mov %rax,%rsi
mov %rdx,%rdi
callq 0x4f <main+79>
lea -0x28(%rbp),%rax
mov %rax,%rsi
lea 0x0(%rip),%rdi
callq 0x62 <main+98>
mov -0x28(%rbp),%rax
test %rax,%rax
je 0xf2 <main+242>
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,-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 0x62 <main+98>
lea 0x0(%rip),%rsi
lea 0x0(%rip),%rdi
callq 0x105 <main+261>
mov %rax,%rdx
mov -0x20(%rbp),%rax
mov %rax,%rsi
mov %rdx,%rdi
callq 0x117 <main+279>
mov %rax,%rdx
mov 0x0(%rip),%rax
mov %rax,%rsi
mov %rdx,%rdi
callq 0x12c <main+300>
mov $0x0,%eax
mov -0x8(%rbp),%rsi
xor %fs:0x28,%rsi
je 0x145 <main+325>
callq 0x145 <main+325>
leaveq
retq		