CS 410 Project One Proficiency Test Template

Explain the functionality of the blocks of assembly code.

“main” function”

Assembly Code Block	Explanation of Functionality
0x0000000000000000 <+0>: push %rbp	push %rbp: Pushes the value of the %rbp register onto the stack.
0x0000000000000001 <+1>: mov %rsp,%rbp	mov %rsp,%rbp: Moves the value of the %rsp register (stack pointer) into the %rbp register (base pointer), establishing a new stack frame.
0x0000000000000004 <+4>: lea 0x0(%rip),%rsi # 0xb <main+11>	lea 0x0(%rip),%rsi: Loads the effective address (address of the next instruction) into the %rsi register.
0x000000000000000b <+11>: lea 0x0(%rip),%rdi # 0x12 <main+18>	lea 0x0(%rip),%rdi: Loads the effective address (address of the next instruction) into the %rdi register.
0x0000000000000012 <+18>: callq 0x17 <main+23>	callq 0x17 <main+23>: Calls the function at address 0x17 (offset from the current instruction).
0x0000000000000017 <+23>: callq 0x1c <main+28>	callq 0x1c <main+28>: Calls the function at address 0x1c (offset from the current instruction).
0x000000000000001c <+28>: mov %eax,0x0(%rip) # 0x22 <main+34>	mov %eax,0x0(%rip): Moves the value in the %eax register to the memory location specified by the address at the next instruction.
0x0000000000000022 <+34>: mov 0x0(%rip),%eax # 0x28 <main+40>	mov 0x0(%rip),%eax: Moves the value stored at the memory location specified by the address at the next instruction into the %eax register.
0x0000000000000028 <+40>: cmp $0x1,%eax	cmp $0x1,%eax: Compares the value in the %eax register with the immediate value 0x1.
0x000000000000002b <+43>: je 0x40 <main+64>	je 0x40 <main+64>: Jumps to address 0x40 (offset from the current instruction) if the previous comparison resulted in equality (zero flag set).
0x000000000000002d <+45>: lea 0x0(%rip),%rsi # 0x34 <main+52>	lea 0x0(%rip),%rsi: Loads the effective address (address of the next instruction) into the %rsi register.
0x0000000000000034 <+52>: lea 0x0(%rip),%rdi # 0x3b <main+59>	lea 0x0(%rip),%rdi: Loads the effective address (address of the next instruction) into the %rdi register.
0x000000000000003b <+59>: callq 0x40 <main+64>	callq 0x40 <main+64>: Calls the function at address 0x40 (offset from the current instruction).
0x0000000000000040 <+64>: mov 0x0(%rip),%eax # 0x46 <main+70>	mov 0x0(%rip),%eax: Moves the value stored at the memory location specified by the address at the next instruction into the %eax register.

0x0000000000000046 <+70>: cmp $0x1,%eax	cmp $0x1,%eax: Compares the value in the %eax register with the immediate value 0x1.
0x0000000000000049 <+73>: je 0x4d <main+77>	je 0x4d <main+77>: Jumps to address 0x4d (offset from the current instruction) if the previous comparison resulted in equality (zero flag set).
0x000000000000004b <+75>: jmp 0x17 <main+23>	jmp 0x17 <main+23>: Unconditional jump to address 0x17 (offset from the current instruction).
0x000000000000004d <+77>: lea 0x0(%rip),%rsi # 0x54 <main+84>	lea 0x0(%rip),%rsi: Loads the effective address (address of the next instruction) into the %rsi register.
0x0000000000000054 <+84>: lea 0x0(%rip),%rdi # 0x5b <main+91>	lea 0x0(%rip),%rdi: Load the effective address of the memory location pointed to by RIP into the %rdi register.
0x000000000000005b <+91>: callq 0x60 <main+96>	callq 0x60 <main+96>: Call the function at memory address 0x60.
0x0000000000000060 <+96>: lea 0x0(%rip),%rsi # 0x67 <main+103>	lea 0x0(%rip),%rsi: Load the effective address of the memory location pointed to by RIP into the %rsi register.
0x0000000000000067 <+103>: lea 0x0(%rip),%rdi # 0x6e <main+110>	lea 0x0(%rip),%rdi: Loads the effective address (address of the next instruction) into the %rdi register.
0x000000000000006e <+110>: callq 0x73 <main+115>	callq 0x73 <main+115>: Calls the function at address 0x73 (offset from the current instruction).
0x0000000000000073 <+115>: lea 0x0(%rip),%rsi # 0x7a <main+122>	lea 0x0(%rip),%rsi: Loads the effective address (address of the next instruction) into the %rsi register.
0x000000000000007a <+122>: lea 0x0(%rip),%rdi # 0x81 <main+129>	lea 0x0(%rip),%rdi: Loads the effective address (address of the next instruction) into the %rdi register.
0x0000000000000081 <+129>: callq 0x86 <main+134>	callq 0x86 <main+134>: Calls the function at address 0x86 (offset from the current instruction).
0x0000000000000086 <+134>: lea 0x0(%rip),%rsi # 0x8d <main+141>	lea 0x0(%rip),%rsi: Loads the effective address (address of the next instruction) into the %rsi register.
0x000000000000008d <+141>: lea 0x0(%rip),%rdi # 0x94 <main+148>	lea 0x0(%rip),%rdi: Loads the effective address (address of the next instruction) into the %rdi register.
0x0000000000000094 <+148>: callq 0x99 <main+153>	callq 0x99 <main+153>: Calls the function at address 0x99 (offset from the current instruction).
0x0000000000000099 <+153>: lea 0x0(%rip),%rsi # 0xa0 <main+160>	lea 0x0(%rip),%rsi: Loads the effective address (address of the next instruction) into the %rsi register.
0x00000000000000a0 <+160>: lea 0x0(%rip),%rdi # 0xa7 <main+167>	lea 0x0(%rip),%rdi: Loads the effective address (address of the next instruction) into the %rdi register.
0x00000000000000a7 <+167>: callq 0xac <main+172>	callq 0xac <main+172>: Calls the function at address 0xac (offset from the current instruction).
0x00000000000000ac <+172>: lea 0x0(%rip),%rsi # 0xb3 <main+179>	lea 0x0(%rip),%rsi: Loads the effective address (address of the next instruction) into the %rsi register.

0x00000000000000b3 <+179>: lea 0x0(%rip),%rdi # 0xba <main+186>	lea 0x0(%rip),%rdi: Loads the effective address (address of the next instruction) into the %rdi register.
0x00000000000000ba <+186>: callq 0xbf <main+191>	callq 0xbf <main+191>: Calls the function at address 0xbf (offset from the current instruction).
0x00000000000000bf <+191>: mov %rax,%rdx	mov %rax,%rdx: Moves the value in the %rax register to the %rdx register.
0x00000000000000c2 <+194>: mov 0x0(%rip),%eax # 0xc8 <main+200>	mov 0x0(%rip),%eax: Moves the value stored at the memory location specified by the address at the next instruction into the %eax register.
0x00000000000000c8 <+200>: mov %eax,%esi	mov %eax,%esi: Moves the value in the %eax register into the %esi register.
0x00000000000000ca <+202>: mov %rdx,%rdi	mov %rdx,%rdi: Moves the value in the %rdx register into the %rdi register.
0x00000000000000cd <+205>: callq 0xd2 <main+210>	callq 0xd2 <main+210>: Calls the function at address 0xd2 (offset from the current instruction).
0x00000000000000d2 <+210>: mov %rax,%rdx	mov %rax,%rdx: Moves the value in the %rax register into the %rdx register.
0x00000000000000d5 <+213>: mov 0x0(%rip),%rax # 0xdc <main+220>	mov 0x0(%rip),%rax: Moves the value stored at the memory location specified by the address at the next instruction into the %rax register.
0x00000000000000dc <+220>: mov %rax,%rsi	mov %rax,%rsi: Moves the value in the %rax register into the %rsi register.
0x00000000000000df <+223>: mov %rdx,%rdi	mov %rdx,%rdi: Moves the value in the %rdx register into the %rdi register.
0x00000000000000e2 <+226>: callq 0xe7 <main+231>	callq 0xe7 <main+231>: Calls the function at address 0xe7 (offset from the current instruction).
0x00000000000000e7 <+231>: mov 0x0(%rip),%eax # 0xed <main+237>	mov 0x0(%rip),%eax: Moves the value stored at the memory location specified by the address at the next instruction into the %eax register.
0x00000000000000ed <+237>: cmp $0x1,%eax	cmp $0x1,%eax: Compares the value in the %eax register with the immediate value 0x1.
0x00000000000000f0 <+240>: jne 0xf9 <main+249>	jne 0xf9 <main+249>: Jumps to address 0xf9 (offset from the current instruction) if the previous comparison resulted in inequality (zero flag not set).
0x00000000000000f2 <+242>: callq 0xf7 <main+247>	callq 0xf7 <main+247>: Calls the function at address 0xf7 (offset from the current instruction).
0x00000000000000f7 <+247>: jmp 0x109 <main+265>	jmp 0x109 <main+265>: Unconditionally jumps to address 0x109 (offset from the current instruction).

0x00000000000000f9 <+249>: mov 0x0(%rip),%eax # 0xff <main+255>	mov 0x0(%rip),%eax: Moves the value stored at the memory location specified by the address at the next instruction into the %eax register.
0x00000000000000ff <+255>: cmp $0x2,%eax	cmp $0x2,%eax: Compares the value in the %eax register with the immediate value 0x2.
0x0000000000000102 <+258>: jne 0x109 <main+265>	jne 0x109 <main+265>: Jumps to address 0x109 (offset from the current instruction) if the previous comparison resulted in inequality (zero flag not set).
0x0000000000000104 <+260>: callq 0x109 <main+265>	callq 0x109 <main+265>: Calls the function at address 0x109 (offset from the current instruction).
0x0000000000000109 <+265>: mov 0x0(%rip),%eax # 0x10f <main+271>	mov 0x0(%rip),%eax: Moves the value stored at the memory location specified by the address at the next instruction into the %eax register.
0x000000000000010f <+271>: cmp $0x3,%eax	cmp $0x3,%eax: Compares the value in the %eax register with the immediate value 0x3.
0x0000000000000112 <+274>: je 0x119 <main+281>	je 0x119 <main+281>: Jumps to address 0x119 (offset from the current instruction) if the previous comparison resulted in equality (zero flag set).
0x0000000000000114 <+276>: jmpq 0x4d <main+77>	jmpq 0x4d <main+77>: Unconditionally jumps to address 0x4d (offset from the current instruction).
0x0000000000000119 <+281>: mov $0x0,%eax	mov $0x0,%eax: Moves the immediate value 0x0 into the %eax register.
0x000000000000011e <+286>: pop %rbp	pop %rbp: Pops the value from the top of the stack into the %rbp register.
0x000000000000011f <+287>: retq	retq: Returns from the function, popping the return address from the stack into the program counter.

ChangeCustomerChoice function

Assembly Code Block	Explanation of Functionality
0x000000000000042d <+0>: push %rbp	push %rbp: Pushes the value of the %rbp register onto the stack.
0x000000000000042e <+1>: mov %rsp,%rbp	mov %rsp,%rbp: Moves the value of the stack pointer (%rsp) into the base pointer (%rbp), setting up a new stack frame.
0x0000000000000431 <+4>: lea 0x0(%rip),%rsi # 0x438 <_Z20ChangeCustomerChoicev+11>	lea 0x0(%rip),%rsi: Computes the effective address of the memory location referenced by the instruction pointer (RIP) and stores it in the %rsi register.
0x0000000000000438 <+11>: lea 0x0(%rip),%rdi # 0x43f <_Z20ChangeCustomerChoicev+18>	lea 0x0(%rip),%rdi: Computes the effective address of the memory location referenced by the instruction pointer (RIP) and stores it in the %rdi register.

0x000000000000043f <+18>: callq 0x444 <_Z20ChangeCustomerChoicev+23>	callq 0x444 <_Z20ChangeCustomerChoicev+23>: Calls the function located at address 0x444, which is 23 bytes ahead of the current instruction.
0x0000000000000444 <+23>: lea 0x0(%rip),%rsi # 0x44b <_Z20ChangeCustomerChoicev+30>	lea 0x0(%rip),%rsi: Computes the effective address of the memory location referenced by the instruction pointer (RIP) and stores it in the %rsi register.
0x000000000000044b <+30>: lea 0x0(%rip),%rdi # 0x452 <_Z20ChangeCustomerChoicev+37>	lea 0x0(%rip),%rdi: Computes the effective address of the memory location referenced by the instruction pointer (RIP) and stores it in the %rdi register.
0x0000000000000452 <+37>: callq 0x457 <_Z20ChangeCustomerChoicev+42>	callq 0x457 <_Z20ChangeCustomerChoicev+42>: Calls the function located at address 0x457, which is 42 bytes ahead of the current instruction.
0x0000000000000457 <+42>: lea 0x0(%rip),%rsi # 0x45e <_Z20ChangeCustomerChoicev+49>	lea 0x0(%rip),%rsi # 0x45e <_Z20ChangeCustomerChoicev+49>: Computes the effective address of the memory location referenced by the instruction pointer (RIP) and stores it in the %rsi register.
0x000000000000045e <+49>: lea 0x0(%rip),%rdi # 0x465 <_Z20ChangeCustomerChoicev+56>	lea 0x0(%rip),%rdi # 0x465 <_Z20ChangeCustomerChoicev+56>: Computes the effective address of the memory location referenced by the instruction pointer (RIP) and stores it in the %rdi register.
0x0000000000000465 <+56>: callq 0x46a <_Z20ChangeCustomerChoicev+61>	callq 0x46a <_Z20ChangeCustomerChoicev+61>: Calls the function located at address 0x46a, which is 61 bytes ahead of the current instruction.
0x000000000000046a <+61>: lea 0x0(%rip),%rsi # 0x471 <_Z20ChangeCustomerChoicev+68>	lea 0x0(%rip),%rsi # 0x471 <_Z20ChangeCustomerChoicev+68>: Computes the effective address of the memory location referenced by the instruction pointer (RIP) and stores it in the %rsi register.
0x0000000000000471 <+68>: lea 0x0(%rip),%rdi # 0x478 <_Z20ChangeCustomerChoicev+75>	lea 0x0(%rip),%rdi # 0x478 <_Z20ChangeCustomerChoicev+75>: Computes the effective address of the memory location referenced by the instruction pointer (RIP) and stores it in the %rdi register.
0x0000000000000478 <+75>: callq 0x47d <_Z20ChangeCustomerChoicev+80>	callq 0x47d <_Z20ChangeCustomerChoicev+80>: Calls the function located at address 0x47d, which is 80 bytes ahead of the current instruction.
0x000000000000047d <+80>: mov 0x0(%rip),%eax # 0x483 <_Z20ChangeCustomerChoicev+86>	mov 0x0(%rip),%eax # 0x483 <_Z20ChangeCustomerChoicev+86>: Moves the value stored at the memory location referenced by the instruction pointer (RIP) into the %eax register.
0x0000000000000483 <+86>: cmp $0x1,%eax	cmp $0x1,%eax: Compares the value in the %eax register with the immediate value 0x1.

0x0000000000000486 <+89>: jne 0x496 <_Z20ChangeCustomerChoicev+105>	jne 0x496 <_Z20ChangeCustomerChoicev+105>: Jumps to address 0x496, which is 105 bytes ahead of the current instruction, if the previous comparison resulted in inequality (zero flag not set).
0x0000000000000488 <+91>: mov 0x0(%rip),%eax # 0x48e <_Z20ChangeCustomerChoicev+97>	mov 0x0(%rip),%eax # 0x48e <_Z20ChangeCustomerChoicev+97>: Moves the value stored at the memory location referenced by the instruction pointer (RIP) into the %eax register.
0x000000000000048e <+97>: mov %eax,0x0(%rip) # 0x494 <_Z20ChangeCustomerChoicev+103>	mov %eax,0x0(%rip) # 0x494 <_Z20ChangeCustomerChoicev+103>: Moves the value in the %eax register to the memory location referenced by the instruction pointer (RIP).
0x0000000000000494 <+103>: jmp 0x4f8 <_Z20ChangeCustomerChoicev+203>	jmp 0x4f8 <_Z20ChangeCustomerChoicev+203>: Jump to address 0x4f8, which is 203 bytes ahead of the current instruction.
0x0000000000000496 <+105>: mov 0x0(%rip),%eax # 0x49c <_Z20ChangeCustomerChoicev+111>	mov 0x0(%rip),%eax # 0x49c <_Z20ChangeCustomerChoicev+111>: Move the value stored at the memory location referenced by the instruction pointer (RIP) into the %eax register.
0x000000000000049c <+111>: cmp $0x2,%eax	cmp $0x2,%eax: Compare the value in the %eax register with the immediate value 0x2.
0x000000000000049f <+114>: jne 0x4af <_Z20ChangeCustomerChoicev+130>	jne 0x4af <_Z20ChangeCustomerChoicev+130>: Jump to address 0x4af, which is 130 bytes ahead of the current instruction, if the previous comparison resulted in inequality (zero flag not set).
0x00000000000004a1 <+116>: mov 0x0(%rip),%eax # 0x4a7 <_Z20ChangeCustomerChoicev+122>	mov 0x0(%rip),%eax # 0x4a7 <_Z20ChangeCustomerChoicev+122>: Move the value stored at the memory location referenced by the instruction pointer (RIP) into the %eax register.
0x00000000000004a7 <+122>: mov %eax,0x0(%rip) # 0x4ad <_Z20ChangeCustomerChoicev+128>	mov %eax,0x0(%rip) # 0x4ad <_Z20ChangeCustomerChoicev+128>: Move the value in the %eax register to the memory location referenced by the instruction pointer (RIP).
0x00000000000004ad <+128>: jmp 0x4f8 <_Z20ChangeCustomerChoicev+203>	jmp 0x4f8 <_Z20ChangeCustomerChoicev+203>: Jump to address 0x4f8, which is 203 bytes ahead of the current instruction.
0x00000000000004af <+130>: mov 0x0(%rip),%eax # 0x4b5 <_Z20ChangeCustomerChoicev+136>	mov 0x0(%rip),%eax # 0x4b5 <_Z20ChangeCustomerChoicev+136>: Move the value stored at the memory location referenced by the instruction pointer (RIP) into the %eax register.
0x00000000000004b5 <+136>: cmp $0x3,%eax	cmp $0x3,%eax: Compare the value in the %eax register with the immediate value 0x3.

0x00000000000004b8 <+139>: jne 0x4c8 <_Z20ChangeCustomerChoicev+155>	jne 0x4c8 <_Z20ChangeCustomerChoicev+155>: Jump to address 0x4c8, which is 155 bytes ahead of the current instruction, if the previous comparison resulted in inequality (zero flag not set).
0x00000000000004ba <+141>: mov 0x0(%rip),%eax # 0x4c0 <_Z20ChangeCustomerChoicev+147>	mov 0x0(%rip),%eax # 0x4c0 <_Z20ChangeCustomerChoicev+147>: Move the value stored at the memory location referenced by the instruction pointer (RIP) into the %eax register.
0x00000000000004c0 <+147>: mov %eax,0x0(%rip) # 0x4c6 <_Z20ChangeCustomerChoicev+153>	mov %eax,0x0(%rip) # 0x4c6 <_Z20ChangeCustomerChoicev+153>: Move the value in the %eax register to the memory location referenced by the instruction pointer (RIP).
0x00000000000004c6 <+153>: jmp 0x4f8 <_Z20ChangeCustomerChoicev+203>	jmp 0x4f8 <_Z20ChangeCustomerChoicev+203>: Jump to address 0x4f8, which is 203 bytes ahead of the current instruction.
0x00000000000004c8 <+155>: mov 0x0(%rip),%eax # 0x4ce <_Z20ChangeCustomerChoicev+161>	mov 0x0(%rip),%eax # 0x4ce <_Z20ChangeCustomerChoicev+161>: Move the value stored at the memory location referenced by the instruction pointer (RIP) into the %eax register.
0x00000000000004ce <+161>: cmp $0x4,%eax	cmp $0x4,%eax: Compare the value in the %eax register with the immediate value 0x4.
0x00000000000004d1 <+164>: jne 0x4e1 <_Z20ChangeCustomerChoicev+180>	jne 0x4e1 <_Z20ChangeCustomerChoicev+180>: Jump to address 0x4e1, which is 180 bytes ahead of the current instruction, if the previous comparison resulted in inequality (zero flag not set).
0x00000000000004d3 <+166>: mov 0x0(%rip),%eax # 0x4d9 <_Z20ChangeCustomerChoicev+172>	mov 0x0(%rip),%eax # 0x4d9 <_Z20ChangeCustomerChoicev+172>: Move the value stored at the memory location referenced by the instruction pointer (RIP) into the %eax register.
0x00000000000004d9 <+172>: mov %eax,0x0(%rip) # 0x4df <_Z20ChangeCustomerChoicev+178>	mov %eax,0x0(%rip) # 0x4df <_Z20ChangeCustomerChoicev+178>: Move the value in the %eax register to the memory location referenced by the instruction pointer (RIP).
0x00000000000004df <+178>: jmp 0x4f8 <_Z20ChangeCustomerChoicev+203>	jmp 0x4f8 <_Z20ChangeCustomerChoicev+203>: Jump to address 0x4f8, which is 203 bytes ahead of the current instruction.
0x00000000000004e1 <+180>: mov 0x0(%rip),%eax # 0x4e7 <_Z20ChangeCustomerChoicev+186>	mov 0x0(%rip),%eax # 0x4e7 <_Z20ChangeCustomerChoicev+186>: Move the value stored at the memory location referenced by the instruction pointer (RIP) into the %eax register.
0x00000000000004e7 <+186>: cmp $0x5,%eax	cmp $0x5,%eax: Compare the value in the %eax register with the immediate value 0x5.

0x00000000000004ea <+189>: jne 0x4f8 <_Z20ChangeCustomerChoicev+203>	jne 0x4f8 <_Z20ChangeCustomerChoicev+203>: Jump to address 0x4f8, which is 203 bytes ahead of the current instruction, if the previous comparison resulted in inequality (zero flag not set).
0x00000000000004ec <+191>: mov 0x0(%rip),%eax # 0x4f2 <_Z20ChangeCustomerChoicev+197>	mov 0x0(%rip),%eax # 0x4f2 <_Z20ChangeCustomerChoicev+197>: Move the value stored at the memory address referenced by the instruction pointer (RIP) into the %eax register.
0x00000000000004f2 <+197>: mov %eax,0x0(%rip) # 0x4f8 <_Z20ChangeCustomerChoicev+203>	mov %eax,0x0(%rip) # 0x4f8 <_Z20ChangeCustomerChoicev+203>: Move the value in the %eax register to the memory location referenced by the instruction pointer (RIP).
0x00000000000004f8 <+203>: nop	nop: No operation.
0x00000000000004f9 <+204>: pop %rbp	pop %rbp: Pop the value from the stack into the %rbp register.
0x00000000000004fa <+205>: retq	retq: Return from the function.

CheckUserPermissonAccess Function

Assembly Code Block	Explanation of Functionality
0x0000000000000120 <+0>: push %rbp	push %rbp: Push the value of the %rbp register onto the stack.
0x0000000000000121 <+1>: mov %rsp,%rbp	mov %rsp,%rbp: Move the value of the %rsp register into the %rbp register.
0x0000000000000124 <+4>: push %rbx	push %rbx: Push the value of the %rbx register onto the stack.
0x0000000000000125 <+5>: sub $0x48,%rsp	sub $0x48,%rsp: Subtract 72 (0x48 in hexadecimal) from the value in the %rsp register.
0x0000000000000129 <+9>: mov %fs:0x28,%rax	mov %fs:0x28,%rax: Move the value stored at the memory address %fs:0x28 into the %rax register.
0x0000000000000132 <+18>: mov %rax,-0x18(%rbp)	mov %rax,-0x18(%rbp): Move the value in the %rax register to the memory location at -0x18 offset from the %rbp register.
0x0000000000000136 <+22>: xor %eax,%eax	xor %eax,%eax: XOR operation on the %eax register with itself, effectively setting it to zero.
0x0000000000000138 <+24>: lea -0x45(%rbp),%rax	lea -0x45(%rbp),%rax: Load effective address, calculating the address -0x45 relative to the %rbp register and storing it in %rax.
0x000000000000013c <+28>: mov %rax,%rdi	mov %rax,%rdi: Move the value in %rax to the %rdi register.
0x000000000000013f <+31>: callq 0x144 <_Z25CheckUserPermissionAccessv+36>	callq 0x144 <_Z25CheckUserPermissionAccessv+36>: Call the function at memory address 0x144.
0x0000000000000144 <+36>: lea -0x45(%rbp),%rdx	lea -0x45(%rbp),%rdx: Load effective address, calculating the address -0x45 relative to the %rbp register and storing it in %rdx.

0x0000000000000148 <+40>: lea -0x40(%rbp),%rax	lea -0x40(%rbp),%rax: Load effective address, calculating the address -0x40 relative to the %rbp register and storing it in %rax.
0x000000000000014c <+44>: lea 0x0(%rip),%rsi # 0x153 <_Z25CheckUserPermissionAccessv+51>	lea 0x0(%rip),%rsi: Load effective address, calculating the address relative to the instruction pointer and storing it in %rsi.
0x0000000000000153 <+51>: mov %rax,%rdi	mov %rax,%rdi: Move the value in %rax to the %rdi register.
0x0000000000000156 <+54>: callq 0x15b <_Z25CheckUserPermissionAccessv+59>	callq 0x15b <_Z25CheckUserPermissionAccessv+59>: Call the function at memory address 0x15b.
0x000000000000015b <+59>: lea -0x45(%rbp),%rax	lea -0x45(%rbp),%rax: Load effective address, calculating the address -0x45 relative to the %rbp register and storing it in %rax.
0x000000000000015f <+63>: mov %rax,%rdi	mov %rax,%rdi: Move the value in %rax to the %rdi register.
0x0000000000000162 <+66>: callq 0x167 <_Z25CheckUserPermissionAccessv+71>	callq 0x167 <_Z25CheckUserPermissionAccessv+71>: Call the function at memory address 0x167.
0x0000000000000167 <+71>: movl $0x0,-0x44(%rbp)	movl $0x0,-0x44(%rbp): Move the immediate value 0x0 (zero) into the memory location at -0x44 offset from the %rbp register.
0x000000000000016e <+78>: lea 0x0(%rip),%rsi # 0x175 <_Z25CheckUserPermissionAccessv+85>	lea 0x0(%rip),%rsi: Load effective address, calculating the address relative to the instruction pointer and storing it in %rsi.
0x0000000000000175 <+85>: lea 0x0(%rip),%rdi # 0x17c <_Z25CheckUserPermissionAccessv+92>	lea 0x0(%rip),%rdi: Load effective address, calculating the address relative to the instruction pointer and storing it in %rdi.
0x000000000000017c <+92>: callq 0x181 <_Z25CheckUserPermissionAccessv+97>	callq 0x181 <_Z25CheckUserPermissionAccessv+97>: Call the function at memory address 0x181.
0x0000000000000181 <+97>: lea 0x0(%rip),%rsi # 0x188 <_Z25CheckUserPermissionAccessv+104>	lea 0x0(%rip),%rsi: Load effective address, calculating the address relative to the instruction pointer and storing it in %rsi.
0x0000000000000188 <+104>: lea 0x0(%rip),%rdi # 0x18f <_Z25CheckUserPermissionAccessv+111>	lea 0x0(%rip),%rdi: Load effective address, calculating the address relative to the instruction pointer and storing it in %rdi.
0x000000000000018f <+111>: callq 0x194 <_Z25CheckUserPermissionAccessv+116>	callq 0x194 <_Z25CheckUserPermissionAccessv+116>: Call the function at memory address 0x194.
0x0000000000000194 <+116>: lea 0x0(%rip),%rsi # 0x19b <_Z25CheckUserPermissionAccessv+123>	lea 0x0(%rip),%rsi: Load effective address, calculating the address relative to the instruction pointer and storing it in %rsi.
0x000000000000019b <+123>: lea 0x0(%rip),%rdi # 0x1a2 <_Z25CheckUserPermissionAccessv+130>	lea 0x0(%rip),%rdi: Load effective address, calculating the address relative to the instruction pointer and storing it in %rdi.
0x00000000000001a2 <+130>: callq 0x1a7 <_Z25CheckUserPermissionAccessv+135>	callq 0x1a7 <_Z25CheckUserPermissionAccessv+135>: Call the function at memory address 0x1a7.
0x00000000000001a7 <+135>: lea -0x40(%rbp),%rax	lea -0x40(%rbp),%rax: Load effective address, calculating the address -0x40 relative to the %rbp register and storing it in %rax.
0x00000000000001ab <+139>: mov %rax,%rsi	mov %rax,%rsi: Move the value in %rax to the %rsi register.

0x00000000000001ae <+142>: lea 0x0(%rip),%rdi # 0x1b5 <_Z25CheckUserPermissionAccessv+149>	lea 0x0(%rip),%rdi: Load effective address, calculating the address relative to the instruction pointer and storing it in %rdi.
0x00000000000001b5 <+149>: callq 0x1ba <_Z25CheckUserPermissionAccessv+154>	callq 0x1ba <_Z25CheckUserPermissionAccessv+154>: Call the function at memory address 0x1ba.
0x00000000000001ba <+154>: lea -0x40(%rbp),%rax	lea -0x40(%rbp),%rax: Load effective address, calculating the address -0x40 relative to the %rbp register and storing it in %rax.
0x00000000000001be <+158>: lea 0x0(%rip),%rsi # 0x1c5 <_Z25CheckUserPermissionAccessv+165>	lea 0x0(%rip),%rsi: Load effective address, calculating the address relative to the instruction pointer and storing it in %rsi.
0x00000000000001c5 <+165>: mov %rax,%rdi	mov %rax,%rdi: Move the value in %rax to the %rdi register.
0x00000000000001c8 <+168>: callq 0x1cd <_Z25CheckUserPermissionAccessv+173>	callq 0x1cd <_Z25CheckUserPermissionAccessv+173>: Call the function at memory address 0x1cd.
0x00000000000001cd <+173>: mov %eax,-0x44(%rbp)	mov %eax,-0x44(%rbp): Move the value in %eax to the memory location at -0x44 offset from the %rbp register.
0x00000000000001d0 <+176>: cmpl $0x0,-0x44(%rbp)	cmpl $0x0,-0x44(%rbp): Compare the value at memory location -0x44 offset from the %rbp register with 0.
0x00000000000001d4 <+180>: jne 0x1dd <_Z25CheckUserPermissionAccessv+189>	jne 0x1dd <_Z25CheckUserPermissionAccessv+189>: Jump to 0x1dd if the previous comparison was not equal (zero flag is not set).
0x00000000000001d6 <+182>: mov $0x1,%ebx	mov $0x1,%ebx: Move the immediate value 0x1 (one) into the %ebx register.
0x00000000000001db <+187>: jmp 0x1e2 <_Z25CheckUserPermissionAccessv+194>	jmp 0x1e2 <_Z25CheckUserPermissionAccessv+194>: Jump to 0x1e2 unconditionally.
0x00000000000001dd <+189>: mov $0x2,%ebx	mov $0x2,%ebx: Move the immediate value 0x2 (two) into the %ebx register.
0x00000000000001e2 <+194>: lea -0x40(%rbp),%rax	lea -0x40(%rbp),%rax: Load effective address, calculating the address -0x40 relative to the %rbp register and storing it in %rax.
0x00000000000001e6 <+198>: mov %rax,%rdi	mov %rax,%rdi: Move the value in %rax to the %rdi register.
0x00000000000001e9 <+201>: callq 0x1ee <_Z25CheckUserPermissionAccessv+206>	callq 0x1ee <_Z25CheckUserPermissionAccessv+206>: Call the function at memory address 0x1ee.
0x00000000000001ee <+206>: mov %ebx,%eax	mov %ebx,%eax: Move the value in %ebx to the %eax register.
0x00000000000001f0 <+208>: mov -0x18(%rbp),%rcx	mov -0x18(%rbp),%rcx: Move the value at memory address -0x18 offset from the %rbp register to the %rcx register.
0x00000000000001f4 <+212>: xor %fs:0x28,%rcx	xor %fs:0x28,%rcx: Perform an XOR operation between the value in the %rcx register and the value at memory address %fs:0x28.
0x00000000000001fd <+221>: je 0x23a <_Z25CheckUserPermissionAccessv+282>	je 0x23a <_Z25CheckUserPermissionAccessv+282>: Jump to 0x23a if the previous comparison resulted in equality (zero flag is set).

0x00000000000001ff <+223>: jmp 0x235 <_Z25CheckUserPermissionAccessv+277>	jmp 0x235 <_Z25CheckUserPermissionAccessv+277>: Unconditional jump to memory address 0x235.
0x0000000000000201 <+225>: mov %rax,%rbx	mov %rax,%rbx: Move the value in %rax to the %rbx register.
0x0000000000000204 <+228>: lea -0x45(%rbp),%rax	lea -0x45(%rbp),%rax: Load effective address, calculating the address -0x45 relative to the %rbp register and storing it in %rax.
0x0000000000000208 <+232>: mov %rax,%rdi	mov %rax,%rdi: Move the value in %rax to the %rdi register.
0x000000000000020b <+235>: callq 0x210 <_Z25CheckUserPermissionAccessv+240>	callq 0x210 <_Z25CheckUserPermissionAccessv+240>: Call the function at memory address 0x210.
0x0000000000000210 <+240>: mov %rbx,%rax	mov %rbx,%rax: Move the value in %rbx to the %rax register.
0x0000000000000213 <+243>: mov %rax,%rdi	mov %rax,%rdi: Move the value in %rax to the %rdi register.
0x0000000000000216 <+246>: callq 0x21b <_Z25CheckUserPermissionAccessv+251>	callq 0x21b <_Z25CheckUserPermissionAccessv+251>: Call the function at memory address 0x21b.
0x000000000000021b <+251>: mov %rax,%rbx	mov %rax,%rbx: Move the value in %rax to the %rbx register.
0x000000000000021e <+254>: lea -0x40(%rbp),%rax	lea -0x40(%rbp),%rax: Load effective address, calculating the address -0x40 relative to the %rbp register and storing it in %rax.
0x0000000000000222 <+258>: mov %rax,%rdi	mov %rax,%rdi: Move the value in %rax to the %rdi register.
0x0000000000000225 <+261>: callq 0x22a <_Z25CheckUserPermissionAccessv+266>	callq 0x22a <_Z25CheckUserPermissionAccessv+266>: Call the function at memory address 0x22a.
0x000000000000022a <+266>: mov %rbx,%rax	mov %rbx,%rax: Move the value in %rbx to the %rax register.
0x000000000000022d <+269>: mov %rax,%rdi	mov %rax,%rdi: Move the value in %rax to the %rdi register.
0x0000000000000230 <+272>: callq 0x235 <_Z25CheckUserPermissionAccessv+277>	callq 0x235 <_Z25CheckUserPermissionAccessv+277>: Call the function at memory address 0x235.
0x0000000000000235 <+277>: callq 0x23a <_Z25CheckUserPermissionAccessv+282>	callq 0x23a <_Z25CheckUserPermissionAccessv+282>: Call the function at memory address 0x23a.
0x000000000000023a <+282>: add $0x48,%rsp	add $0x48,%rsp: Add 0x48 to the %rsp register.
0x000000000000023e <+286>: pop %rbx	pop %rbx: Pop the top value from the stack and store it in the %rbx register.
0x000000000000023f <+287>: pop %rbp	pop %rbp: Pop the top value from the stack and store it in the %rbp register.
0x0000000000000240 <+288>: retq	retq: Return from the function.

DisplayInfo Function

Assembly Code Block	Explanation of Functionality
0x0000000000000241 <+0>: push %rbp	push %rbp: Push the value of the %rbp register onto the stack.
0x0000000000000242 <+1>: mov %rsp,%rbp	mov %rsp,%rbp: Move the value of the %rsp register into the %rbp register.

0x0000000000000245 <+4>: lea 0x0(%rip),%rsi # 0x24c <_Z11DisplayInfov+11>	lea 0x0(%rip),%rsi: Load the effective address of the memory location pointed to by RIP into the %rsi register.
0x000000000000024c <+11>: lea 0x0(%rip),%rdi # 0x253 <_Z11DisplayInfov+18>	lea 0x0(%rip),%rdi: Load the effective address of the memory location pointed to by RIP into the %rdi register.
0x0000000000000253 <+18>: callq 0x258 <_Z11DisplayInfov+23>	callq 0x258 <_Z11DisplayInfov+23>: Call the function at memory address 0x258.
0x0000000000000258 <+23>: mov %rax,%rdx	mov %rax,%rdx: Move the value in the %rax register to the %rdx register.
0x000000000000025b <+26>: mov 0x0(%rip),%rax # 0x262 <_Z11DisplayInfov+33>	mov 0x0(%rip),%rax: Move the value at the memory address pointed to by RIP to the %rax register.
0x0000000000000262 <+33>: mov %rax,%rsi	mov %rax,%rsi: Move the value in the %rax register to the %rsi register.
0x0000000000000265 <+36>: mov %rdx,%rdi	mov %rdx,%rdi: Move the value in the %rdx register to the %rdi register.
0x0000000000000268 <+39>: callq 0x26d <_Z11DisplayInfov+44>	callq 0x26d <_Z11DisplayInfov+44>: Call the function at memory address 0x26d.
0x000000000000026d <+44>: lea 0x0(%rip),%rsi # 0x274 <_Z11DisplayInfov+51>	lea 0x0(%rip),%rsi: Load the effective address of the memory location pointed to by RIP into the %rsi register.
0x0000000000000274 <+51>: lea 0x0(%rip),%rdi # 0x27b <_Z11DisplayInfov+58>	lea 0x0(%rip),%rdi: Load the effective address of the memory location pointed to by RIP into the %rdi register.
0x000000000000027b <+58>: callq 0x280 <_Z11DisplayInfov+63>	callq 0x280 <_Z11DisplayInfov+63>: Call the function at memory address 0x280.
0x0000000000000280 <+63>: lea 0x0(%rip),%rsi # 0x287 <_Z11DisplayInfov+70>	lea 0x0(%rip),%rsi: Load the effective address of the memory location pointed to by RIP into the %rsi register.
0x0000000000000287 <+70>: mov %rax,%rdi	mov %rax,%rdi: Move the value in the %rax register to the %rdi register.
0x000000000000028a <+73>: callq 0x28f <_Z11DisplayInfov+78>	callq 0x28f <_Z11DisplayInfov+78>: Call the function at memory address 0x28f.
0x000000000000028f <+78>: lea 0x0(%rip),%rsi # 0x296 <_Z11DisplayInfov+85>	lea 0x0(%rip),%rsi: Load the effective address of the memory location pointed to by RIP into the %rsi register.
0x0000000000000296 <+85>: mov %rax,%rdi	mov %rax,%rdi: Move the value in the %rax register to the %rdi register.
0x0000000000000299 <+88>: callq 0x29e <_Z11DisplayInfov+93>	callq 0x29e <_Z11DisplayInfov+93>: Call the function at memory address 0x29e.
0x000000000000029e <+93>: mov %rax,%rdx	mov %rax,%rdx: Move the value in the %rax register to the %rdx register.
0x00000000000002a1 <+96>: mov 0x0(%rip),%eax # 0x2a7 <_Z11DisplayInfov+102>	mov 0x0(%rip),%eax: Move the value at the memory address pointed to by RIP to the %eax register.

0x00000000000002a7 <+102>: mov %eax,%esi	mov %eax,%esi: Move the value in the %eax register to the %esi register.
0x00000000000002a9 <+104>: mov %rdx,%rdi	mov %rdx,%rdi: Move the value in the %rdx register to the %rdi register.
0x00000000000002ac <+107>: callq 0x2b1 <_Z11DisplayInfov+112>	callq 0x2b1 <_Z11DisplayInfov+112>: Call the function at memory address 0x2b1.
0x00000000000002b1 <+112>: mov %rax,%rdx	mov %rax,%rdx: Move the value in the %rax register to the %rdx register.
0x00000000000002b4 <+115>: mov 0x0(%rip),%rax # 0x2bb <_Z11DisplayInfov+122>	mov 0x0(%rip),%rax: Move the value at the memory address pointed to by RIP to the %rax register.
0x00000000000002bb <+122>: mov %rax,%rsi	mov %rax,%rsi: Move the value in the %rax register to the %rsi register.
0x00000000000002be <+125>: mov %rdx,%rdi	mov %rdx,%rdi: Move the value in the %rdx register to the %rdi register.
0x00000000000002c1 <+128>: callq 0x2c6 <_Z11DisplayInfov+133>	callq 0x2c6 <_Z11DisplayInfov+133>: Call the function at memory address 0x2c6.
0x00000000000002c6 <+133>: lea 0x0(%rip),%rsi # 0x2cd <_Z11DisplayInfov+140>	lea 0x0(%rip),%rsi: Load the effective address of the memory location pointed to by RIP into the %rsi register.
0x00000000000002cd <+140>: lea 0x0(%rip),%rdi # 0x2d4 <_Z11DisplayInfov+147>	lea 0x0(%rip),%rdi: Load the effective address of the memory location pointed to by RIP into the %rdi register.
0x00000000000002d4 <+147>: callq 0x2d9 <_Z11DisplayInfov+152>	callq 0x2d9 <_Z11DisplayInfov+152>: Call the function at memory address 0x2d9.
0x00000000000002d9 <+152>: lea 0x0(%rip),%rsi # 0x2e0 <_Z11DisplayInfov+159>	lea 0x0(%rip),%rsi: Load the effective address of the memory location pointed to by RIP into the %rsi register.
0x00000000000002e0 <+159>: mov %rax,%rdi	mov %rax,%rdi: Move the value in the %rax register to the %rdi register.
0x00000000000002e3 <+162>: callq 0x2e8 <_Z11DisplayInfov+167>	callq 0x2e8 <_Z11DisplayInfov+167>: Call the function at memory address 0x2e8.
0x00000000000002e8 <+167>: lea 0x0(%rip),%rsi # 0x2ef <_Z11DisplayInfov+174>	lea 0x0(%rip),%rsi: Load the effective address of the memory location pointed to by RIP into the %rsi register.
0x00000000000002ef <+174>: mov %rax,%rdi	mov %rax,%rdi: Move the value in the %rax register to the %rdi register.
0x00000000000002f2 <+177>: callq 0x2f7 <_Z11DisplayInfov+182>	callq 0x2f7 <_Z11DisplayInfov+182>: Call the function at memory address 0x2f7.
0x00000000000002f7 <+182>: mov %rax,%rdx	mov %rax,%rdx: Move the value in the %rax register to the %rdx register.
0x00000000000002fa <+185>: mov 0x0(%rip),%eax # 0x300 <_Z11DisplayInfov+191>	mov 0x0(%rip),%eax: Move the value at the memory address pointed to by RIP to the %eax register.

0x0000000000000300 <+191>: mov %eax,%esi	mov %eax,%esi: Move the value in the %eax register to the %esi register.
0x0000000000000302 <+193>: mov %rdx,%rdi	mov %rdx,%rdi: Move the value in the %rdx register to the %rdi register.
0x0000000000000305 <+196>: callq 0x30a <_Z11DisplayInfov+201>	callq 0x30a <_Z11DisplayInfov+201>: Call the function at memory address 0x30a.
0x000000000000030a <+201>: mov %rax,%rdx	mov %rax,%rdx: Move the value in the %rax register to the %rdx register.
0x000000000000030d <+204>: mov 0x0(%rip),%rax # 0x314 <_Z11DisplayInfov+211>	mov 0x0(%rip),%rax: Move the value located at the memory address pointed to by RIP offset by 0 bytes into the %rax register.
0x0000000000000314 <+211>: mov %rax,%rsi	mov %rax,%rsi: Move the value in the %rax register to the %rsi register.
0x0000000000000317 <+214>: mov %rdx,%rdi	mov %rdx,%rdi: Move the value in the %rdx register to the %rdi register.
0x000000000000031a <+217>: callq 0x31f <_Z11DisplayInfov+222>	callq 0x31f <_Z11DisplayInfov+222>: Call the function located at memory address 0x31f, which is likely the DisplayInfo function or a similar routine.
0x000000000000031f <+222>: lea 0x0(%rip),%rsi # 0x326 <_Z11DisplayInfov+229>	lea 0x0(%rip),%rsi: Load the effective address of the memory location pointed to by RIP into the %rsi register.
0x0000000000000326 <+229>: lea 0x0(%rip),%rdi # 0x32d <_Z11DisplayInfov+236>	lea 0x0(%rip),%rdi: Load the effective address of the memory location pointed to by RIP into the %rdi register.
0x000000000000032d <+236>: callq 0x332 <_Z11DisplayInfov+241>	callq 0x332 <_Z11DisplayInfov+241>: Call the function located at memory address 0x332.
0x0000000000000332 <+241>: lea 0x0(%rip),%rsi # 0x339 <_Z11DisplayInfov+248>	lea 0x0(%rip),%rsi: Load the effective address of the memory location pointed to by RIP into the %rsi register.
0x0000000000000339 <+248>: mov %rax,%rdi	mov %rax,%rdi: Move the value in the %rax register to the %rdi register.
0x000000000000033c <+251>: callq 0x341 <_Z11DisplayInfov+256>	callq 0x341 <_Z11DisplayInfov+256>: Call the function located at memory address 0x341.
0x0000000000000341 <+256>: lea 0x0(%rip),%rsi # 0x348 <_Z11DisplayInfov+263>	lea 0x0(%rip),%rsi: Load the effective address of the memory location pointed to by RIP into the %rsi register.
0x0000000000000348 <+263>: mov %rax,%rdi	mov %rax,%rdi: Move the value in the %rax register to the %rdi register.
0x000000000000034b <+266>: callq 0x350 <_Z11DisplayInfov+271>	callq 0x350 <_Z11DisplayInfov+271>: Call the function located at memory address 0x350.
0x0000000000000350 <+271>: mov %rax,%rdx	mov %rax,%rdx: Move the value in the %rax register to the %rdx register.

0x0000000000000353 <+274>: mov 0x0(%rip),%eax # 0x359 <_Z11DisplayInfov+280>	mov 0x0(%rip),%eax: Move the value located at the memory address pointed to by the RIP register offset by 0 bytes into the %eax register.
0x0000000000000359 <+280>: mov %eax,%esi	mov %eax,%esi: Move the value in the %eax register to the %esi register.
0x000000000000035b <+282>: mov %rdx,%rdi	mov %rdx,%rdi: Move the value in the %rdx register to the %rdi register.
0x000000000000035e <+285>: callq 0x363 <_Z11DisplayInfov+290>	callq 0x363 <_Z11DisplayInfov+290>: Call the function located at memory address 0x363, likely the DisplayInfo function or a similar routine.
0x0000000000000363 <+290>: mov %rax,%rdx	mov %rax,%rdx: Move the value returned by the function call (presumably in %rax) to the %rdx register.
0x0000000000000366 <+293>: mov 0x0(%rip),%rax # 0x36d <_Z11DisplayInfov+300>	0x0000000000000366 <+293>: mov 0x0(%rip),%rax # 0x36d <_Z11DisplayInfov+300>
0x000000000000036d <+300>: mov %rax,%rsi	mov %rax,%rsi: Move the value in the %rax register to the %rsi register.
0x0000000000000370 <+303>: mov %rdx,%rdi	mov %rdx,%rdi: Move the value in the %rdx register to the %rdi register.
0x0000000000000373 <+306>: callq 0x378 <_Z11DisplayInfov+311>	callq 0x378 <_Z11DisplayInfov+311>: Call the function at memory address 0x378.
0x0000000000000378 <+311>: lea 0x0(%rip),%rsi # 0x37f <_Z11DisplayInfov+318>	lea 0x0(%rip),%rsi: Load the effective address of the memory location pointed to by RIP into the %rsi register.
0x000000000000037f <+318>: lea 0x0(%rip),%rdi # 0x386 <_Z11DisplayIn---Typefov+325>	lea 0x0(%rip),%rdi: Load the effective address of the memory location pointed to by RIP into the %rdi register.
0x0000000000000386 <+325>: callq 0x38b <_Z11DisplayInfov+330>	callq 0x38b <_Z11DisplayInfov+330>: Call the function at memory address 0x38b.
0x000000000000038b <+330>: lea 0x0(%rip),%rsi # 0x392 <_Z11DisplayInfov+337>	lea 0x0(%rip),%rsi: Load the effective address of the memory location pointed to by RIP into the %rsi register.
0x0000000000000392 <+337>: mov %rax,%rdi	mov %rax,%rdi: Move the value in the %rax register to the %rdi register.
0x0000000000000395 <+340>: callq 0x39a <_Z11DisplayInfov+345>	callq 0x39a <_Z11DisplayInfov+345>: Call the function at memory address 0x39a.
0x000000000000039a <+345>: lea 0x0(%rip),%rsi # 0x3a1 <_Z11DisplayInfov+352>	lea 0x0(%rip),%rsi: Load the effective address of the memory location pointed to by RIP into the %rsi register.
0x00000000000003a1 <+352>: mov %rax,%rdi	mov %rax,%rdi: Move the value in the %rax register to the %rdi register.
0x00000000000003a4 <+355>: callq 0x3a9 <_Z11DisplayInfov+360>	callq 0x3a9 <_Z11DisplayInfov+360>: Call the function at memory address 0x3a9.
0x00000000000003a9 <+360>: mov %rax,%rdx	mov %rax,%rdx: Move the value in the %rax register to the %rdx register.

0x00000000000003ac <+363>: mov 0x0(%rip),%eax # 0x3b2 <_Z11DisplayInfov+369>	mov 0x0(%rip),%eax: Move the value at the memory address pointed to by RIP to the %eax register.
0x00000000000003b2 <+369>: mov %eax,%esi	mov %eax,%esi: Move the value in the %eax register to the %esi register.
0x00000000000003b4 <+371>: mov %rdx,%rdi	mov %rdx,%rdi: Move the value in the %rdx register to the %rdi register.
0x00000000000003b7 <+374>: callq 0x3bc <_Z11DisplayInfov+379>	callq 0x3bc <_Z11DisplayInfov+379>: Call the function at memory address 0x3bc.
0x00000000000003bc <+379>: mov %rax,%rdx	mov %rax,%rdx: Move the value in the %rax register to the %rdx register.
0x00000000000003bf <+382>: mov 0x0(%rip),%rax # 0x3c6 <_Z11DisplayInfov+389>	mov 0x0(%rip),%rax: Move the value at the memory address pointed to by RIP to the %rax register.
0x00000000000003c6 <+389>: mov %rax,%rsi	mov %rax,%rsi: Move the value in the %rax register to the %rsi register.
0x00000000000003c9 <+392>: mov %rdx,%rdi	mov %rdx,%rdi: Move the value in the %rdx register to the %rdi register.
0x00000000000003cc <+395>: callq 0x3d1 <_Z11DisplayInfov+400>	callq 0x3d1 <_Z11DisplayInfov+400>: Call the function at memory address 0x3d1.
0x00000000000003d1 <+400>: lea 0x0(%rip),%rsi # 0x3d8 <_Z11DisplayInfov+407>	lea 0x0(%rip),%rsi: Load the effective address of the memory location pointed to by RIP into the %rsi register.
0x00000000000003d8 <+407>: lea 0x0(%rip),%rdi # 0x3df <_Z11DisplayInfov+414>	lea 0x0(%rip),%rdi: Load the effective address of the memory location pointed to by RIP into the %rdi register.
0x00000000000003df <+414>: callq 0x3e4 <_Z11DisplayInfov+419>	callq 0x3e4 <_Z11DisplayInfov+419>: Call the function at memory address 0x3e4.
0x00000000000003e4 <+419>: lea 0x0(%rip),%rsi # 0x3eb <_Z11DisplayInfov+426>	lea 0x0(%rip),%rsi: Load the effective address of the memory location pointed to by RIP into the %rsi register.
0x00000000000003eb <+426>: mov %rax,%rdi	mov %rax,%rdi: Move the value in the %rax register to the %rdi register.
0x00000000000003ee <+429>: callq 0x3f3 <_Z11DisplayInfov+434>	callq 0x3f3 <_Z11DisplayInfov+434>: Call the function at memory address 0x3f3.
0x00000000000003f3 <+434>: lea 0x0(%rip),%rsi # 0x3fa <_Z11DisplayInfov+441>	lea 0x0(%rip),%rsi: Load the effective address of the memory location pointed to by RIP into the %rsi register.
0x00000000000003fa <+441>: mov %rax,%rdi	mov %rax,%rdi: Move the value in the %rax register to the %rdi register.
0x00000000000003fd <+444>: callq 0x402 <_Z11DisplayInfov+449>	callq 0x402 <_Z11DisplayInfov+449>: Call the function at memory address 0x402.
0x0000000000000402 <+449>: mov %rax,%rdx	mov %rax,%rdx: Move the value in the %rax register to the %rdx register.

0x0000000000000405 <+452>: mov 0x0(%rip),%eax # 0x40b <_Z11DisplayInfov+458>	mov 0x0(%rip),%eax: Move the value at the memory address pointed to by RIP to the %eax register.
0x000000000000040b <+458>: mov %eax,%esi	mov %eax,%esi: Move the value in the %eax register to the %esi register.
0x000000000000040d <+460>: mov %rdx,%rdi	mov %rdx,%rdi: Move the value in the %rdx register to the %rdi register.
0x0000000000000410 <+463>: callq 0x415 <_Z11DisplayInfov+468>	callq 0x415 <_Z11DisplayInfov+468>: Call the function at memory address 0x415.
0x0000000000000415 <+468>: mov %rax,%rdx	mov %rax,%rdx: Move the value in the %rax register to the %rdx register.
0x0000000000000418 <+471>: mov 0x0(%rip),%rax # 0x41f <_Z11DisplayInfov+478>	mov 0x0(%rip),%rax: Move the value at the memory address pointed to by RIP to the %rax register.
0x000000000000041f <+478>: mov %rax,%rsi	mov %rax,%rsi: Move the value in the %rax register to the %rsi register.
0x0000000000000422 <+481>: mov %rdx,%rdi	mov %rdx,%rdi: Move the value in the %rdx register to the %rdi register.
0x0000000000000425 <+484>: callq 0x42a <_Z11DisplayInfov+489>	callq 0x42a <_Z11DisplayInfov+489>: Call the function at memory address 0x42a.
0x000000000000042a <+489>: nop	nop: No operation.
0x000000000000042b <+490>: pop %rbp	pop %rbp: Restore the value of the base pointer from the stack.
0x000000000000042c <+491>: retq	retq: Return from the function.