# CS 410 Project One Proficiency Test Template

## Explain the functionality of the blocks of assembly code.

### “main” function”

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  lea 0x5eb(%rip),%rsi # 0x1440  lea 0x201244(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  callq 0xf6a <\_Z25CheckUserPermissionAccessv> | Push register rbp to the stack  Move resp to register rbp  Load effective address of 0x5eb of rip to register rsi  Load effective address of 0x2144 of rip to register rdi  Call instruction to call the CheckUserPermission function aid authentication of the user password |
| mov %eax,0x201494(%rip) # 0x202300 <answer>  mov 0x20148e(%rip),%eax # 0x202300 <answer>  cmp $0x1,%eax  je 0xe8a <main+64>  lea 0x5f2(%rip),%rsi # 0x1470  lea 0x20121b(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov 0x201470(%rip),%eax # 0x202300 <answer>  cmp $0x1,%eax  je 0xe97 <main+77>  jmp 0xe61 <main+23> | Move eax to 0x1494 of rip. This will be the first user input for username  Move 0x148e of rip to register eax. This will be the user input for password.  After the checkUserPermission function is called, the main function will compare the output of the function to that of eax (the entered password.  Jump if equal instruction used to jump to CheckUserPermission function if output and eax are equal.  Load effective address of 0x5f2 of rip to rsi. This will be a callback to the Checkuserpermission function  This block is the same as before where it checks if the jump equal instruction. If equal the String output of “Invalid Password” will be displayed. The CheckUserPermission function will then be accessed again.  Unconditional jump to the next portion of the main function to progress the program |
| lea 0x5f6(%rip),%rsi # 0x1494  lea 0x2011fb(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0xc90  <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x5ff(%rip),%rsi # 0x14b0  lea 0x2011e8(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x614(%rip),%rsi # 0x14d8  lea 0x2011d5(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x625(%rip),%rsi # 0x14fc  lea 0x2011c2(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> | Load effective addresses of string instructions of the program to display the string “What would you like to do?”  Load effective address of register values to access string instruction that will output the first menu option of "DISPLAY the client list (enter 1)"  Load effective address of register values to access string instruction that will output the first menu option of "CHANGE a client's choice (enter 2)"  Load effective address of register values to access string instruction that will output the first menu option of "EXIT the program (enter 3)" |
| lea 0x20140a(%rip),%rsi # 0x2022f4 <choice>  lea 0x2012cf(%rip),%rdi # 0x2021c0 <\_ZSt3cin@@GLIBCXX\_3.4>  callq 0xc60 <\_ZNSirsERi@plt> | Load effective address of 140a of rip to register rsi. This will be the cin call in the program to gather user input  The lea call of 12cf of rip to register rdi will be the user input to the program. The call instruction calls this to allow the user to input the answer |
| lea 0x61d(%rip),%rsi # 0x151a  lea 0x20119c(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> | Load effective address of 61d of rip to rsi  Load effective address 119c of rip to rsi  Call instruction to call the string “You chose” and the user input choice to the output |
| mov %rax,%rdx  mov 0x2013e2(%rip),%eax # 0x2022f4 <choice>  mov %eax,%esi  mov %rdx,%rdi  callq 0xd00 <\_ZNSolsEi@plt> | Move rax to register rdx  Move 13e2 of rip to register eax. This will be the user input choice  Move eax to register esi  Move rdx to register rdi  Call instruction to output the choice the user entered |
| mov 0x2013bd(%rip),%eax # 0x2022f4 <choice>  cmp $0x1,%eax  jne 0xf43 <main+249>  callq 0x108b <\_Z11DisplayInfov>  jmp 0xf53 <main+265> | Move 13bd of rip to register eax. This value will be the value of the user input  Compare register eax to 1.  If not equal to 1, jump the program to the next block of code  If equal to 1, call instruction to call the DisplayInfo function |
| mov 0x2013ab(%rip),%eax # 0x2022f4 <choice>  cmp $0x2,%eax  jne 0xf53 <main+265>  callq 0x1277 <\_Z20ChangeCustomerChoicev> | Move 13ab of rip to register eax. This will be the value of the user input  Compare register eax value to 2.  If not equal to 2, jump the program to the next block of code  If equal to 2, call instruction to call the ChangeCustomerChoice function |
| mov 0x20139b(%rip),%eax # 0x2022f4 <choice>  cmp $0x3,%eax  je 0xf63 <main+281>  jmpq 0xe97 <main+77> | Move 139b of rip to register eax. This will represent the value of the user input  Compare register eax to 3.  If equal to three, then jump to quit to exit the program |

### ChangeCustomerChoice function

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  lea 0x346(%rip),%rsi # 0x15c8  lea 0x200e17(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> | Push register rbp to the stack  Move rsp to register rbp  Load effective address 346 of rip to rsi. This load will loa the instruction for the string "Enter the number of the client that you wish to change". This string will be output via the call instruction |
| lea 0x201063(%rip),%rsi # 0x2022f8 <changechoice>  lea 0x200f24(%rip),%rdi # 0x2021c0 <\_ZSt3cin@@GLIBCXX\_3.4>  callq 0xc60 <\_ZNSirsERi@plt> | Load effective address of 1063 of rip to register rsi. This load will be used to gather user input for the change\_choice |
| lea 0x358(%rip),%rsi # 0x1600  lea 0x200df1(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> | Load effective address of 358 of rip to register rsi. This will load the instruction for the string.  The call instruction will output the string "Please enter the client's new service choice (1 = Brokerage, 2 = Retirement)" |
| lea 0x201041(%rip),%rsi # 0x2022fc <newservice>  lea 0x200efe(%rip),%rdi # 0x2021c0  <\_ZSt3cin@@GLIBCXX\_3.4>  callq 0xc60 <\_ZNSirsERi@plt> | Load effective address of 1041 of rip to register rsi. This load will take place in order to gather the user input for the new\_service variable. |
| mov 0x20102b(%rip),%eax # 0x2022f8 <changechoice>  cmp $0x1,%eax  jne 0x12e0 <\_Z20ChangeCustomerChoicev+105>  mov 0x201024(%rip),%eax # 0x2022fc <newservice>  mov %eax,0x200d82(%rip) # 0x202060 <num1>  jmp 0x1342 <\_Z20ChangeCustomerChoicev+203> | Move 102b of rip to register eax. This value stored to eax is the user input value  Compare register eax value to 1.  If not equal to 1, jump to the next conditional block of assembly code.  If equal to 1, then move 1024 of rip to register eax. This will be the value of the user input for new\_service  Move eax to rip at d82 bits. This is storing the value of eax to this location. This is the equivalent of modifying an element in an array in C++. We are replacing the value.  Unconditional jump to the main function, exiting the function |
| mov 0x201012(%rip),%eax # 0x2022f8 <changechoice>  cmp $0x2,%eax  jne 0x12f9 <\_Z20ChangeCustomerChoicev+130>  mov 0x20100b(%rip),%eax # 0x2022fc <newservice>  mov %eax,0x200d6d(%rip) # 0x202064 <num2>  jmp 0x1342 <\_Z20ChangeCustomerChoicev+203> | Move 1012 of rip to register eax. This value stored to eax is the user input value  Compare register eax value to 2.  If not equal to 2, jump to the next conditional block of assembly code.  If equal to 2, then move 100b of rip to register eax. This will be the value of the user input for new\_service  Move eax to rip at d82 bits. This is storing the value of eax to this location. This is the equivalent of modifying an element in an array in C++. We are replacing the value.  Unconditional jump to the main function, exiting the function |
| mov 0x200ff9(%rip),%eax # 0x2022f8 <changechoice>  cmp $0x3,%eax  jne 0x1312 <\_Z20ChangeCustomerChoicev+155>  mov 0x200ff2(%rip),%eax # 0x2022fc <newservice>  mov %eax,0x200d58(%rip) # 0x202068 <num3>  jmp 0x1342 <\_Z20ChangeCustomerChoicev+203> | Move 0ff9 of rip to register eax. This value stored to eax is the user input value  Compare register eax value to 3.  If not equal to 3, jump to the next conditional block of assembly code.  If equal to 3, then move off2 of rip to register eax. This will be the value of the user input for new\_service  Move eax to rip at d58 bits. This is storing the value of eax to this location. This is the equivalent of modifying an element in an array in C++. We are replacing the value.  Unconditional jump to the main function, exiting the function |
| mov 0x200fe0(%rip),%eax # 0x2022f8 <changechoice>  cmp $0x4,%eax  jne 0x132b <\_Z20ChangeCustomerChoicev+180>  mov 0x200fd9(%rip),%eax # 0x2022fc <newservice>  mov %eax,0x200d43(%rip) # 0x20206c <num4>  jmp 0x1342 <\_Z20ChangeCustomerChoicev+203> | Move 0fe0 of rip to register eax. This value stored to eax is the user input value  Compare register eax value to 4.  If not equal to 4, jump to the next conditional block of assembly code.  If equal to 4, then move ofd9 of rip to register eax. This will be the value of the user input for new\_service  Move eax to rip at d43 bits. This is storing the value of eax to this location. This is the equivalent of modifying an element in an array in C++. We are replacing the value.  Unconditional jump to the main function, exiting the function |
| mov 0x200fc7(%rip),%eax # 0x2022f8 <changechoice>  cmp $0x5,%eax  jne 0x1342 <\_Z20ChangeCustomerChoicev+203>  mov 0x200fc0(%rip),%eax # 0x2022fc <newservice>  mov %eax,0x200d2e(%rip) # 0x202070 <num5>  nop | Move 0fe0 of rip to register eax. This value stored to eax is the user input value  Compare register eax value to 5.  If not equal to 5, and no other conditions were met in the for loop of the assembly code, the function will jump to the main function  If equal to 4, then move ofc0 of rip to register eax. This will be the value of the user input for new\_service  Move eax to rip at 0d2e bits. This is storing the value of eax to this location. This is the equivalent of modifying an element in an array in C++. We are replacing the value.  A no operation instruction is used, due to there being no other element to jump to in the arrays used |
| pop %rbp  retq | Pop the values from register rbp  Return from the function to the main function |

### CheckUserPermissonAccess Function

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  push %rbx  sub $0x48,%rsp  mov %fs:0x28,%rax  mov %rax,-0x18(%rbp)  xor %eax,%eax  lea -0x45(%rbp),%rax  mov %rax,%rdi  callq 0xd20 <\_ZNSaIcEC1Ev@plt> | Push register rbp to the stack  Move rsp to register rbp  Push register rbx to the stack  Subtract 48 from register rsp  Move 0x28 to register rax  Move rax to register rbp at 18 bits  Xor instruction used to set eax to 0 or 1. By comparison this happens with the instruction  Load effective address rbp at 45 bits to register rax  Move rax to register rdi.  This section of the assembly code is setting up the variables for the function and the intitial setup of the function before parts are intitialized |
| lea -0x45(%rbp),%rdx  lea -0x40(%rbp),%rax  lea 0x588(%rip),%rsi # 0x1525  mov %rax,%rdi  callq 0xce0 <\_ZNSt7\_\_cxx1112basic\_stringIcSt11char\_traitsIcESaIcEEC1EPKcRKS3\_@plt>  lea -0x45(%rbp),%rax  mov %rax,%rdi  0xcb0 <\_ZNSaIcED1Ev@plt>  movl $0x0,-0x44(%rbp)  lea 0x567(%rip),%rsi # 0x1526  lea 0x2010da(%rip),%rdi # 0x2020a0  <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0xc90  <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> | Load effective address of rbp at 45 bits to register rdx  Load effective address of rbp at 40 bits to register rax  Load effective address of 0x588 of rip to register rsi. This will load the string value instruction to the register.  Call the instruction to output the string “Enter your username: “.  Load effective address of rbp at 45 bits to rax  Move rax to register rdi  Move 0x0 to rbp at 44 bits  Load effective address of rip at 0x567 to rsi. This will bring the string instruction into register rsi  Load effective address of rip at 10da bits to register rdi  Call string instruction to output to the system |
| lea 0x20130e(%rip),%rsi # 0x2022e0 <username>  lea 0x2011e7(%rip),%rdi # 0x2021c0 <\_ZSt3cin@@GLIBCXX\_3.4>  callq 0xc40  <\_ZStrsIcSt11char\_traitsIcEERSt13basic\_istreamIT\_T0\_ES6\_PS3\_@plt> | Load effective address of 0x130e of rip to register rsi. This will be the variable username.  Load effective address od 0x11e7 of rip to register rdi. This will be the instruction for gathering user input and setting it to the variable username  Call the instruction to gather user input and set it to username |
| lea 0x558(%rip),%rsi # 0x153d  lea 0x2010b4(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> | Load effective address of 558 of rip to register rsi. This points to the string instruction in the system.  Load effective address of 10b4 of rip to register rdi  Call instruction to output the string loaded in the output “ Enter your password: “. |
| lea -0x40(%rbp),%rax  mov %rax,%rsi  lea 0x2011c1(%rip),%rdi # 0x2021c0 <\_ZSt3cin@@GLIBCXX\_3.4>  callq 0xcd0 <\_ZStrsIcSt11char\_traitsIcESaIcEERSt13basic\_istreamIT\_T0\_ES7\_RNSt7\_\_cxx1112basic\_stringIS4\_S5\_T1\_EE@plt> | Load effective address of rbp at 40 bits to register rax  Move rax to register rsi  Load effective address of 11c1 of rip to rdi. This will load the instruction for user input. The user input will be set to variable password  Call instruction to load in the instruction to gather user input for password |
| lea -0x40(%rbp),%rax  lea 0x545(%rip),%rsi # 0x1554  mov %rax,%rdi  callq 0xc50 <\_ZNKSt7\_\_cxx1112basic\_stringIcSt11char\_traitsIcESaIcEE7compareEPKc@plt> | Load effective address of rbp at 40 bits to register rax  Load effective address of rip at 0x545 bits to register rsi  Move rax to register rdi  Call instruction for the string “123”. This is the password embedded into the program. This string will be stored as the valid password in array passwords. |
| mov %eax,-0x44(%rbp)  cmpl $0x0,-0x44(%rbp)  jne 0x1027 <\_Z25CheckUserPermissionAccessv+189>  mov $0x1,%ebx  jmp 0x102c <\_Z25CheckUserPermissionAccessv+194>  mov $0x2,%ebx  lea -0x40(%rbp),%rax  mov %rax,%rdi  callq 0xc70 <\_ZNSt7\_\_cxx1112basic\_stringIcSt11char\_traitsIcESaIcEED1Ev@plt> | This segment compares the user input for the password. This in a sense shows the iteration of int I through the usernames array. It sets the integer I to the user input if it equal to that of one of the elements of usernames array. |
| mov %ebx,%eax  mov -0x18(%rbp),%rcx  xor %fs:0x28,%rcx  je 0x1084 <\_Z25CheckUserPermissionAccessv+282>  jmp 0x107f <\_Z25CheckUserPermissionAccessv+277> | This is the iteration of variable j through the passwords array. If the passwords variables is equal to one of the elements of the array j will be set to the password variable from the user input |
| mov %rax,%rbx  lea -0x45(%rbp),%rax  mov %rax,%rdi  callq 0xcb0 <\_ZNSaIcED1Ev@plt>  mov %rbx,%rax  mov %rax,%rdi  callq 0xd10 <\_Unwind\_Resume@plt>  mov %rax,%rbx  lea -0x40(%rbp),%rax  mov %rax,%rdi  callq 0xc70 <\_ZNSt7\_\_cxx1112basic\_stringIcSt11char\_traitsIcESaIcEED1Ev@plt>  mov %rbx,%rax  mov %rax,%rdi  callq 0xd10 <\_Unwind\_Resume@plt>  callq 0xcc0 <\_\_stack\_chk\_fail@plt | This section checks to see if variable I and j are equal. If variable I and j are equal then the boolean match will be set to true. This is set to true and will be used later in the main to authenticate the username and password that is entered. |

### DisplayInfo Function

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  lea 0x4c2(%rip),%rsi # 0x1558  lea 0x201003(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> | Push rbp to the stack  Move rsp to register rbp  Load effective address of 0x4c2 of rip to register rsi. This will load the string instruction that is stored in the program.  Load effective address 1003 od rip to register rdi. This will load the string instruction stored within the program  Call instruction to output the string instructions loaded to output the message "Client's Name Service Selected (1. Brokerage 2. Retirement) “ |
| mov %rax,%rdx  mov 0x200f24(%rip),%rax # 0x201fd0  mov %rax,%rsi  mov %rdx,%rdi  callq 0xca0 <\_ZNSolsEPFRSoS\_E@plt>  lea 0x4de(%rip),%rsi # 0x159c  0x200fdb(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x200f3f(%rip),%rsi # 0x202010 <name1>  mov %rax,%rdi  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x4c0(%rip),%rsi # 0x15a0  mov %rax,%rdi  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov 0x200f6f(%rip),%eax # 0x202060 <num1>  mov %eax,%esi  mov %rdx,%rdi  callq 0xd00 <\_ZNSolsEi@plt> | Move rax to register rdi  Move 0x0f24 of rip to register rax  Move rax to register rsi  Move rdx to register rdi  Call instruction to load int I into the program to output the number of the client.  Call instruction to load the iteration of I into the output. It will be each element of the array.  Load 0x4de of rip into register rsi. This is loading the iteration of I in array name. Ths will be used in the next call instruction  Move rax into register rdi  Call instruction load each element of the array name. This output to the console “ 1. “ and element of the array name.  Load effective address of 0x4c0 of rip into register rsi  Move rax in to register rdi  Call instruction to output the string that follows the previous call instruction to look like “ I , name[i], “ selected option “.  Move rax to register rdi  Move 0x0f6f of rip to register eax. This will select the element from the choice array. This uses the same iteration of I that the name array used.  Move eax to register esi  Move rdx to register rdi  Call instruction to output the element of choice[i] to the console. An example of this output would be “ 1. Bob Jones selected option 1”. |
| mov %rax,%rdx  mov 0x200ecb(%rip),%rax # 0x201fd0  mov %rax,%rsi  mov %rdx,%rdi  callq 0xca0 <\_ZNSolsEPFRSoS\_E@plt>  lea 0x49b(%rip),%rsi # 0x15b2  lea 0x200f82(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x200ef6(%rip),%rsi # 0x202020 <name2>  mov %rax,%rdi  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x467(%rip),%rsi # 0x15a0  mov %rax,%rdi  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov 0x200f1a(%rip),%eax # 0x202064 <num2>  mov %eax,%esi  mov %rdx,%rdi  callq 0xd00 <\_ZNSolsEi@plt> | Move rax to register rdx  Move 0x0ecb of rip to register rax  Move rax to register rsi  Move rdx to register rdi  Call instruction to load int I into the program to output the number of the client.  Call instruction to load the iteration of I into the output. It will be each element of the array.  Load 0x0ef6 of rip into register rsi. This is loading the iteration of I in array name. This will be used in the next call instruction  Move rax into register rdi  Call instruction load each element of the array name. This output to the console “ 1. “ and element of the array name.  Load effective address of 0x467 of rip into register rsi  Move rax in to register rdi  Call instruction to output the string that follows the previous call instruction to look like “ I , name[i], “ selected option “.  Move rax to register rdi  Move 0x0f1a of rip to register eax. This will select the element from the choice array. This uses the same iteration of I that the name array used.  Move eax to register esi  Move rdx to register rdi  Call instruction to output the element of choice[i] to the console. An example of this output would be “ 1. Bob Jones selected option 1”. |
| mov %rax,%rdx  mov 0x200e72(%rip),%rax # 0x201fd0  mov %rax,%rsi  mov %rdx,%rdi  callq 0xca0 <\_ZNSolsEPFRSoS\_E@plt>  lea 0x446(%rip),%rsi # 0x15b6  lea 0x200f29(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x200ead(%rip),%rsi # 0x202030 <name3>  mov %rax,%rdi  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x40e(%rip),%rsi # 0x15a0  mov %rax,%rdi  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov 0x200ec5(%rip),%eax # 0x202068 <num3>  mov %eax,%esi  mov %rdx,%rdi  callq 0xd00 <\_ZNSolsEi@plt> | Move rax to register rdx  Move 0x0e72 of rip to register rax  Move rax to register rsi  Move rdx to register rdi  Call instruction to load int I into the program to output the number of the client.  Call instruction to load the iteration of I into the output. It will be each element of the array.  Load 0x0ead of rip into register rsi. This is loading the iteration of I in array name. This will be used in the next call instruction  Move rax into register rdi  Call instruction load each element of the array name. This output to the console “ 1. “ and element of the array name.  Load effective address of 0x40e of rip into register rsi  Move rax in to register rdi  Call instruction to output the string that follows the previous call instruction to look like “ I , name[i], “ selected option “.  Move rax to register rdi  Move 0x0ec5 of rip to register eax. This will select the element from the choice array. This uses the same iteration of I that the name array used.  Move eax to register esi  Move rdx to register rdi  Call instruction to output the element of choice[i] to the console. An example of this output would be “ 1. Bob Jones selected option 1”. |
| mov %rax,%rdx  mov 0x200e19(%rip),%rax # 0x201fd0  mov %rax,%rsi  mov %rdx,%rdi  callq 0xca0 <\_ZNSolsEPFRSoS\_E@plt>  lea 0x3f1(%rip),%rsi # 0x15ba  lea 0x200ed0(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x200e64(%rip),%rsi # 0x202040 <name4>  mov %rax,%rdi  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x3b5(%rip),%rsi # 0x15a0  mov %rax,%rdi  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov 0x200e70(%rip),%eax # 0x20206c <num4>  mov %eax,%esi  mov %rdx,%rdi  callq 0xd00 <\_ZNSolsEi@plt> | Move rax to register rdx  Move 0x0e19 of rip to register rax  Move rax to register rsi  Move rdx to register rdi  Call instruction to load int I into the program to output the number of the client.  Call instruction to load the iteration of I into the output. It will be each element of the array.  Load 0x0e64 of rip into register rsi. This is loading the iteration of I in array name. This will be used in the next call instruction  Move rax into register rdi  Call instruction load each element of the array name. This output to the console “ 1. “ and element of the array name.  Load effective address of 0x3b5 of rip into register rsi  Move rax in to register rdi  Call instruction to output the string that follows the previous call instruction to look like “ I , name[i], “ selected option “.  Move rax to register rdi  Move 0x0e70 of rip to register eax. This will select the element from the choice array. This uses the same iteration of I that the name array used.  Move eax to register esi  Move rdx to register rdi  Call instruction to output the element of choice[i] to the console. An example of this output would be “ 1. Bob Jones selected option 1”. |
| mov %rax,%rdx  mov 0x200dc0(%rip),%rax # 0x201fd0  mov %rax,%rsi  mov %rdx,%rdi  callq 0xca0 <\_ZNSolsEPFRSoS\_E@plt>  lea 0x39c(%rip),%rsi # 0x15be  lea 0x200e77(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x200e1b(%rip),%rsi # 0x202050 <name5>  mov %rax,%rdi  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x35c(%rip),%rsi # 0x15a0  mov %rax,%rdi  callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov 0x200e1b(%rip),%eax # 0x202070 <num5>  mov %eax,%esi  mov %rdx,%rdi  callq 0xd00 <\_ZNSolsEi@plt> | Move rax to register rdx  Move 0x0dc0 of rip to register rax  Move rax to register rsi  Move rdx to register rdi  Call instruction to load int I into the program to output the number of the client.  Call instruction to load the iteration of I into the output. It will be each element of the array.  Load 0x0e1b of rip into register rsi. This is loading the iteration of I in array name. This will be used in the next call instruction  Move rax into register rdi  Call instruction load each element of the array name. This output to the console “ 1. “ and element of the array name.  Load effective address of 0x35c of rip into register rsi  Move rax in to register rdi  Call instruction to output the string that follows the previous call instruction to look like “ I , name[i], “ selected option “.  Move rax to register rdi  Move 0x0e1b of rip to register eax. This will select the element from the choice array. This uses the same iteration of I that the name array used.  Move eax to register esi  Move rdx to register rdi  Call instruction to output the element of choice[i] to the console. An example of this output would be “ 1. Bob Jones selected option 1”. |