# CS 410 Project One Proficiency Test – Kerrian Offermann

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

### “main” function”

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| 0x0000000000000e4a <+0>: push %rbp | Call the function that will print an output on screen. |
| 0x0000000000000e4b <+1>: mov %rsp,%rbp |
| 0x0000000000000e4e <+4>: lea 0x5eb(%rip),%rsi # 0x1440 |
| 0x0000000000000e55 <+11>: lea 0x201244(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4> |
| 0x0000000000000e5c <+18>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x0000000000000e61 <+23>: callq 0xf6a <\_Z25CheckUserPermissionAccessv> | Call the CheckUserPermissionAccess() function. When the user enters their answer, move the answer to %eax and then compare it to “1”.  If the value is equal to 1 then call the print the output.  Enter username  Invalid password. Please try again. |
| 0x0000000000000e66 <+28>: mov %eax,0x201494(%rip) # 0x202300 <answer> |
| 0x0000000000000e6c <+34>: mov 0x20148e(%rip),%eax # 0x202300 <answer> |
| 0x0000000000000e72 <+40>: cmp $0x1,%eax |
| 0x0000000000000e75 <+43>: je 0xe8a <main+64> |
| 0x0000000000000e77 <+45>: lea 0x5f2(%rip),%rsi # 0x1470 |
| 0x0000000000000e7e <+52>: lea 0x20121b(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4> |
| 0x0000000000000e85 <+59>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x0000000000000e8a <+64>: mov 0x201470(%rip),%eax # 0x202300 <answer> | Move answer to eax and compare to 1. If the value is equal to 1, then the loop will repeat and the print function will be called to print the output on screen. |
| 0x0000000000000e90 <+70>: cmp $0x1,%eax |
| 0x0000000000000e93 <+73>: je 0xe97 <main+77> |
| 0x0000000000000e95 <+75>: jmp 0xe61 <main+23> |
| 0x0000000000000e97 <+77>: lea 0x5f6(%rip),%rsi # 0x1494 |
| 0x0000000000000e9e <+84>: lea 0x2011fb(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4> |
| 0x0000000000000ea5 <+91>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x0000000000000eaa <+96>: lea 0x5ff(%rip),%rsi # 0x14b0 | Point 0x5FF to rsi and 0x2011e8 to rdi for printing operations to the screen. |
| 0x0000000000000eb1 <+103>: lea 0x2011e8(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4> |
| 0x0000000000000eb8 <+110>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x0000000000000ebd <+115>: lea 0x614(%rip),%rsi # 0x14d8 | Point 0x614 to rsi and 0x2011d5 to rdi for printing operations to the screen. |
| 0x0000000000000ec4 <+122>: lea 0x2011d5(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4> |
| 0x0000000000000ecb <+129>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x0000000000000ed0 <+134>: lea 0x625(%rip),%rsi # 0x14fc | Point 0x625 to rsi and 0x2011c2 to rdi for printing operations to the screen. |
| 0x0000000000000ed7 <+141>: lea 0x2011c2(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4> |
| 0x0000000000000ede <+148>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x0000000000000ee3 <+153>: lea 0x20140a(%rip),%rsi # 0x2022f4 <choice> | Point 0x20140a to rsi (user’s choice) and 0x2012cf to rdi input (“cin >>”) |
| 0x0000000000000eea <+160>: lea 0x2012cf(%rip),%rdi # 0x2021c0 <\_ZSt3cin@@GLIBCXX\_3.4> |
| 0x0000000000000ef1 <+167>: callq 0xc60 <\_ZNSirsERi@plt> (operator>>(%int)) |
| 0x0000000000000ef6 <+172>: lea 0x61d(%rip),%rsi # 0x151a | Point 0x61d to rsi and 0x20119c to rdi for printing operations to the screen. |
| 0x0000000000000efd <+179>: lea 0x20119c(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4> |
| 0x0000000000000f04 <+186>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x0000000000000f09 <+191>: mov %rax,%rdx | Move user’s choice to register %eax then %eax to %esi. Also move rax to rdx and rdx to rdi. Call the operator “<<” which is often used in “cout <<” |
| 0x0000000000000f0c <+194>: mov 0x2013e2(%rip),%eax # 0x2022f4 <choice> |
| 0x0000000000000f12 <+200>: mov %eax,%esi |
| 0x0000000000000f14 <+202>: mov %rdx,%rdi |
| 0x0000000000000f17 <+205>: callq 0xd00 <\_ZNSolsEi@plt> operator<<(int) |
| 0x0000000000000f1c <+210>: mov %rax,%rdx | Call the “<<” operator which is often in “cout <<” |
| 0x0000000000000f1f <+213>: mov 0x2010aa(%rip),%rax # 0x201fd0 |
| 0x0000000000000f26 <+220>: mov %rax,%rsi |
| 0x0000000000000f29 <+223>: mov %rdx,%rdi |
| 0x0000000000000f2c <+226>: callq 0xca0 <\_ZNSolsEPFRSoS\_E@plt> operator<< |
| 0x0000000000000f31 <+231>: mov 0x2013bd(%rip),%eax # 0x2022f4 <choice> | Compare the choice entered (in eax) to 1. If the choice is equal to 1 then call the “DisplayInfo” function. |
| 0x0000000000000f37 <+237>: cmp $0x1,%eax |
| 0x0000000000000f3a <+240>: jne 0xf43 <main+249> |
| 0x0000000000000f3c <+242>: callq 0x108b <\_Z11DisplayInfov> |
| 0x0000000000000f41 <+247>: jmp 0xf53 <main+265> | Restart the loop. Compare the choice entered by user to 2. If the choice is equal to 2 then call the “Change Customer Choice” function. |
| 0x0000000000000f43 <+249>: mov 0x2013ab(%rip),%eax # 0x2022f4 <choice> |
| 0x0000000000000f49 <+255>: cmp $0x2,%eax |
| 0x0000000000000f4c <+258>: jne 0xf53 <main+265> |
| 0x0000000000000f4e <+260>: callq 0x1277 <\_Z20ChangeCustomerChoicev> |
| 0x0000000000000f53 <+265>: mov 0x20139b(%rip),%eax # 0x2022f4 <choice> | Compare user choice to value 3. If it is equal to 3 then then exit the program. |
| 0x0000000000000f59 <+271>: cmp $0x3,%eax |
| 0x0000000000000f5c <+274>: je 0xf63 <main+281> |
| 0x0000000000000f5e <+276>: jmpq 0xe97 <main+77> |
| 0x0000000000000f63 <+281>: mov $0x0,%eax |
| 0x0000000000000f68 <+286>: pop %rbp |
| 0x0000000000000f69 <+287>: retq |

### ChangeCustomerChoice function

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| 0x0000000000001277 <+0>: push %rbp | Call the function that will print an output on screen. |
| 0x0000000000001278 <+1>: mov %rsp,%rbp |
| 0x000000000000127b <+4>: lea 0x346(%rip),%rsi # 0x15c8 |
| 0x0000000000001282 <+11>: lea 0x200e17(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4> |
| 0x0000000000001289 <+18>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x000000000000128e <+23>: lea 0x201063(%rip),%rsi # 0x2022f8 <changechoice> | Point to addresses that allow user to input change of choice. Call the “>>” operator which is used when using “cin >>” |
| 0x0000000000001295 <+30>: lea 0x200f24(%rip),%rdi # 0x2021c0 <\_ZSt3cin@@GLIBCXX\_3.4> |
| 0x000000000000129c <+37>: callq 0xc60 <\_ZNSirsERi@plt> |
| 0x00000000000012a1 <+42>: lea 0x358(%rip),%rsi # 0x1600 | Point to cout address and prepare to print to screen |
| 0x00000000000012a8 <+49>: lea 0x200df1(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4> |
| 0x00000000000012af <+56>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x00000000000012b4 <+61>: lea 0x201041(%rip),%rsi # 0x2022fc <newservice> | Point to addresses that allow user to input new service. Call the “>>” operator which is used when using “cin >>” |
| 0x00000000000012bb <+68>: lea 0x200efe(%rip),%rdi # 0x2021c0 <\_ZSt3cin@@GLIBCXX\_3.4> |
| 0x00000000000012c2 <+75>: callq 0xc60 <\_ZNSirsERi@plt> |
| 0x00000000000012c7 <+80>: mov 0x20102b(%rip),%eax # 0x2022f8 <changechoice> | Move user’s input from “change choice” to %eax register then compare to “1”. If user’s choice is equal to 1 then call “Change Customer Choice” function.  Move new service to %eax and then to num1 (Bob Jones).  Jump to “Change Customer Choice”.  Move change choice to “eax” |
| 0x00000000000012cd <+86>: cmp $0x1,%eax |
| 0x00000000000012d0 <+89>: jne 0x12e0 <\_Z20ChangeCustomerChoicev+105> |
| 0x00000000000012d2 <+91>: mov 0x201024(%rip),%eax # 0x2022fc <newservice> |
| 0x00000000000012d8 <+97>: mov %eax,0x200d82(%rip) # 0x202060 <num1> |
| 0x00000000000012de <+103>: jmp 0x1342 <\_Z20ChangeCustomerChoicev+203> |
| 0x00000000000012e0 <+105>: mov 0x201012(%rip),%eax # 0x2022f8 <changechoice> |
| 0x00000000000012e6 <+111>: cmp $0x2,%eax | Compare “changechoice” entry to “2”. If the choice is equal to “2” then move new service to eax and then eax to num2 (Sarah Davis).  Jump to “Change Customer Choice” and move “change choice” to eax. |
| 0x00000000000012e9 <+114>: jne 0x12f9 <\_Z20ChangeCustomerChoicev+130> |
| 0x00000000000012eb <+116>: mov 0x20100b(%rip),%eax # 0x2022fc <newservice> |
| 0x00000000000012f1 <+122>: mov %eax,0x200d6d(%rip) # 0x202064 <num2> |
| 0x00000000000012f7 <+128>: jmp 0x1342 <\_Z20ChangeCustomerChoicev+203> |
| 0x00000000000012f9 <+130>: mov 0x200ff9(%rip),%eax # 0x2022f8 <changechoice> |
| 0x00000000000012ff <+136>: cmp $0x3,%eax | Compare “changechoice” entry to “3”. If the choice is equal to “3” then move new service to eax and then eax to num3 (Amy Friendly).  Jump to “Change Customer Choice” and move “change choice” to eax. |
| 0x0000000000001302 <+139>: jne 0x1312 <\_Z20ChangeCustomerChoicev+155> |
| 0x0000000000001304 <+141>: mov 0x200ff2(%rip),%eax # 0x2022fc <newservice> |
| 0x000000000000130a <+147>: mov %eax,0x200d58(%rip) # 0x202068 <num3> |
| 0x0000000000001310 <+153>: jmp 0x1342 <\_Z20ChangeCustomerChoicev+203> |
| 0x0000000000001312 <+155>: mov 0x200fe0(%rip),%eax # 0x2022f8 <changechoice> |
| 0x0000000000001318 <+161>: cmp $0x4,%eax | Compare “changechoice” entry to “4”. If the choice is equal to “4” then move new service to eax and then eax to num4 (Johnny Smith).  Jump to “Change Customer Choice” function and move “change choice” to eax. |
| 0x000000000000131b <+164>: jne 0x132b <\_Z20ChangeCustomerChoicev+180> |
| 0x000000000000131d <+166>: mov 0x200fd9(%rip),%eax # 0x2022fc <newservice> |
| 0x0000000000001323 <+172>: mov %eax,0x200d43(%rip) # 0x20206c <num4> |
| 0x0000000000001329 <+178>: jmp 0x1342 <\_Z20ChangeCustomerChoicev+203> |
| 0x000000000000132b <+180>: mov 0x200fc7(%rip),%eax # 0x2022f8 <changechoice> |
| 0x0000000000001331 <+186>: cmp $0x5,%eax | Compare “changechoice” entry to “5”. If the choice is equal to “5” then move new service to eax and then eax to num5 (Carol Spears).  Jump to “Change Customer Choice” function and move “change choice” to eax. |
| 0x0000000000001334 <+189>: jne 0x1342 <\_Z20ChangeCustomerChoicev+203> |
| 0x0000000000001336 <+191>: mov 0x200fc0(%rip),%eax # 0x2022fc <newservice> |
| 0x000000000000133c <+197>: mov %eax,0x200d2e(%rip) # 0x202070 <num5> |
| 0x0000000000001342 <+203>: nop | Return to main menu once choice is made. |
| 0x0000000000001343 <+204>: pop %rbp |
| 0x0000000000001344 <+205>: retq |

### CheckUserPermissonAccess Function

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| 0x0000000000000f6a <+0>: push %rbp | Push %rbp to the front of the stack before moving rsp to rbp. Push rbx next in the stack.  From here, a stack-guard value (%fs:0x28) is used along with xor to check and compare values.  At the end, a call is made to “allocator ()” to allocate memory management in the code. |
| 0x0000000000000f6b <+1>: mov %rsp,%rbp |
| 0x0000000000000f6e <+4>: push %rbx |
| 0x0000000000000f6f <+5>: sub $0x48,%rsp |
| 0x0000000000000f73 <+9>: mov %fs:0x28,%rax |
| 0x0000000000000f7c <+18>: mov %rax,-0x18(%rbp) |
| 0x0000000000000f80 <+22>: xor %eax,%eax |
| 0x0000000000000f82 <+24>: lea -0x45(%rbp),%rax |
| 0x0000000000000f86 <+28>: mov %rax,%rdi |
| 0x0000000000000f89 <+31>: callq 0xd20 <\_ZNSaIcEC1Ev@plt> |
| 0x0000000000000f8e <+36>: lea -0x45(%rbp),%rdx | Lea points different register addresses to other registers before rax is move to rdi.  A call is made to allocate memory. |
| 0x0000000000000f92 <+40>: lea -0x40(%rbp),%rax |
| 0x0000000000000f96 <+44>: lea 0x588(%rip),%rsi # 0x1525 |
| 0x0000000000000f9d <+51>: mov %rax,%rdi |
| 0x0000000000000fa0 <+54>: callq 0xce0 <\_ZNSt7\_\_cxx1112basic\_stringIcSt11char\_traitsIcESaIcEEC1EPKcRKS3\_@plt> |
| 0x0000000000000fa5 <+59>: lea -0x45(%rbp),%rax | Point -45%rbp to rax then move rax to rdi. Call allocator again to allocate memory. |
| 0x0000000000000fa9 <+63>: mov %rax,%rdi |
| 0x0000000000000fac <+66>: callq 0xcb0 <\_ZNSaIcED1Ev@plt> |
| 0x0000000000000fb1 <+71>: movl $0x0,-0x44(%rbp) | Move value of 0 to -44%rbp. Point the address of 0x567(%rip) to rsi.  The print function is called to print the details of “cout” to the screen. |
| 0x0000000000000fb8 <+78>: lea 0x567(%rip),%rsi # 0x1526 |
| 0x0000000000000fbf <+85>: lea 0x2010da(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4> |
| 0x0000000000000fc6 <+92>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x0000000000000fcb <+97>: lea 0x20130e(%rip),%rsi # 0x2022e0 <username> | The address of username of pointed to “cin” where user will enter their username. |
| 0x0000000000000fd2 <+104>: lea 0x2011e7(%rip),%rdi # 0x2021c0 <\_ZSt3cin@@GLIBCXX\_3.4> |
| 0x0000000000000fd9 <+111>: callq 0xc40 <\_ZStrsIcSt11char\_traitsIcEERSt13basic\_istreamIT\_T0\_ES6\_PS3\_@plt> |
| 0x0000000000000fde <+116>: lea 0x558(%rip),%rsi # 0x153d | Print function is called to print to screen the results of cout. |
| 0x0000000000000fe5 <+123>: lea 0x2010b4(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4> |
| 0x0000000000000fec <+130>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x0000000000000ff1 <+135>: lea -0x40(%rbp),%rax | -40%rbp is moved to rax then rax is moved to rsi. The address for “cin” is called for user input. |
| 0x0000000000000ff5 <+139>: mov %rax,%rsi |
| 0x0000000000000ff8 <+142>: lea 0x2011c1(%rip),%rdi # 0x2021c0 <\_ZSt3cin@@GLIBCXX\_3.4> |
| 0x0000000000000fff <+149>: callq 0xcd0 <\_ZStrsIcSt11char\_traitsIcESaIcEERSt13basic\_istreamIT\_T0\_ES7\_RNSt7\_\_cxx1112basic\_stringIS4\_S5\_T1\_EE@plt> |
| 0x0000000000001004 <+154>: lea -0x40(%rbp),%rax | -40%rbp is moved to rax then rax is moved to rsi again. Rax is moved to rdi. A function called to start doing comparison. |
| 0x0000000000001008 <+158>: lea 0x545(%rip),%rsi # 0x1554 |
| 0x000000000000100f <+165>: mov %rax,%rdi |
| 0x0000000000001012 <+168>: callq 0xc50 <\_ZNKSt7\_\_cxx1112basic\_stringIcSt11char\_traitsIcESaIcEE7compareEPKc@plt> |
| 0x0000000000001017 <+173>: mov %eax,-0x44(%rbp) | %eax is moved to -44%rbp. A comparison is made between 0 and the value in -0x44. If that value is equal to 0 then the Check User Permission Access function is called.  1 is moved to ebx before the instruction jumps to the Check User Permission Access function.  The value 2 is now moved to %ebx.  Call is made to an allocator for characters. |
| 0x000000000000101a <+176>: cmpl $0x0,-0x44(%rbp) |
| 0x000000000000101e <+180>: jne 0x1027 <\_Z25CheckUserPermissionAccessv+189> |
| 0x0000000000001020 <+182>: mov $0x1,%ebx |
| 0x0000000000001025 <+187>: jmp 0x102c <\_Z25CheckUserPermissionAccessv+194> |
| 0x0000000000001027 <+189>: mov $0x2,%ebx |
| 0x000000000000102c <+194>: lea -0x40(%rbp),%rax |
| 0x0000000000001030 <+198>: mov %rax,%rdi |
| 0x0000000000001033 <+201>: callq 0xc70 <\_ZNSt7\_\_cxx1112basic\_stringIcSt11char\_traitsIcESaIcEED1Ev@plt> |
| 0x0000000000001038 <+206>: mov %ebx,%eax | %ebx is moved to eax before -18%rbp’s value is moved to rcx. When rcx is compared to stack guard value in %fs:0x28, xor compares them. If they are equal then the Check User Permission Access function is called.  A jump is made to Check User Permission Access function.  The allocate memory function is called. |
| 0x000000000000103a <+208>: mov -0x18(%rbp),%rcx |
| 0x000000000000103e <+212>: xor %fs:0x28,%rcx |
| 0x0000000000001047 <+221>: je 0x1084 <\_Z25CheckUserPermissionAccessv+282> |
| 0x0000000000001049 <+223>: jmp 0x107f <\_Z25CheckUserPermissionAccessv+277> |
| 0x000000000000104b <+225>: mov %rax,%rbx |
| 0x000000000000104e <+228>: lea -0x45(%rbp),%rax |
| 0x0000000000001052 <+232>: mov %rax,%rdi |
| 0x0000000000001055 <+235>: callq 0xcb0 <\_ZNSaIcED1Ev@plt> |
| 0x000000000000105a <+240>: mov %rbx,%rax | %rbx is moved to %rax before %rax is moved to %rdi. A call is made to a function called Unwind\_Resume. |
| 0x000000000000105d <+243>: mov %rax,%rdi |
| 0x0000000000001060 <+246>: callq 0xd10 <\_Unwind\_Resume@plt> |
| 0x0000000000001065 <+251>: mov %rax,%rbx | A call is made to our character allocator for strings. |
| 0x0000000000001068 <+254>: lea -0x40(%rbp),%rax |
| 0x000000000000106c <+258>: mov %rax,%rdi |
| 0x000000000000106f <+261>: callq 0xc70 <\_ZNSt7\_\_cxx1112basic\_stringIcSt11char\_traitsIcESaIcEED1Ev@plt> |
| 0x0000000000001074 <+266>: mov %rbx,%rax | Another call is made to Unwind\_Resume after rbx is moved to rax and rax is moved to rbi.  A call is made to check if there is a stack check failure.  Changes are made to registers rsp and rbx where rbp is popped from the top of the stack.  Return to menu |
| 0x0000000000001077 <+269>: mov %rax,%rdi |
| 0x000000000000107a <+272>: callq 0xd10 <\_Unwind\_Resume@plt> |
| 0x000000000000107f <+277>: callq 0xcc0 <\_\_stack\_chk\_fail@plt> |
| 0x0000000000001084 <+282>: add $0x48,%rsp |
| 0x0000000000001088 <+286>: pop %rbx |
| 0x0000000000001089 <+287>: pop %rbp |
| 0x000000000000108a <+288>: retq |

### DisplayInfo Function

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| 0x000000000000108b <+0>: push %rbp | Push %rbp to the front of the stack then move %rsp into %rbp.  Call the function that will print “cout” on to the screen. |
| 0x000000000000108c <+1>: mov %rsp,%rbp |
| 0x000000000000108f <+4>: lea 0x4c2(%rip),%rsi # 0x1558 |
| 0x0000000000001096 <+11>: lea 0x201003(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4> |
| 0x000000000000109d <+18>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x00000000000010a2 <+23>: mov %rax,%rdx | Call the operator “<<” for the cout function to print the characters for cout. |
| 0x00000000000010a5 <+26>: mov 0x200f24(%rip),%rax # 0x201fd0 |
| 0x00000000000010ac <+33>: mov %rax,%rsi |
| 0x00000000000010af <+36>: mov %rdx,%rdi |
| 0x00000000000010b2 <+39>: callq 0xca0 <\_ZNSolsEPFRSoS\_E@plt> |
| 0x00000000000010b7 <+44>: lea 0x4de(%rip),%rsi # 0x159c | Call another “cout” function to be printed to screen. |
| 0x00000000000010be <+51>: lea 0x200fdb(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4> |
| 0x00000000000010c5 <+58>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x00000000000010ca <+63>: lea 0x200f3f(%rip),%rsi # 0x202010 <name1> | Point to the memory address containing name1 (Bob Jones) and call the output function to print the charcters. |
| 0x00000000000010d1 <+70>: mov %rax,%rdi |
| 0x00000000000010d4 <+73>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x00000000000010d9 <+78>: lea 0x4c0(%rip),%rsi # 0x15a0 | Call the operator << (used for cout) and the output stream that will print characters. |
| 0x00000000000010e0 <+85>: mov %rax,%rdi |
| 0x00000000000010e3 <+88>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x00000000000010e8 <+93>: mov %rax,%rdx | Move option 1 (Bob Jones) to eax then to rsi. After moving rdx to rdi, call operator “<<” . |
| 0x00000000000010eb <+96>: mov 0x200f6f(%rip),%eax # 0x202060 <num1> |
| 0x00000000000010f1 <+102>: mov %eax,%esi |
| 0x00000000000010f3 <+104>: mov %rdx,%rdi |
| 0x00000000000010f6 <+107>: callq 0xd00 <\_ZNSolsEi@plt> |
| 0x00000000000010fb <+112>: mov %rax,%rdx | After moving registers, call operator “<<” for the output stream that will print outcome. |
| 0x00000000000010fe <+115>: mov 0x200ecb(%rip),%rax # 0x201fd0 |
| 0x0000000000001105 <+122>: mov %rax,%rsi |
| 0x0000000000001108 <+125>: mov %rdx,%rdi |
| 0x000000000000110b <+128>: callq 0xca0 <\_ZNSolsEPFRSoS\_E@plt> |
| 0x0000000000001110 <+133>: lea 0x49b(%rip),%rsi # 0x15b2 | Point to the address for “cout” and then call the “<<” operator to print the characters in cout. |
| 0x0000000000001117 <+140>: lea 0x200f82(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4> |
| 0x000000000000111e <+147>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x0000000000001123 <+152>: lea 0x200ef6(%rip),%rsi # 0x202020 <name2> | Point name2 (Sarah Davis) to %rsi. Move %rax to %rdi. Call the output stream and the operator <<. |
| 0x000000000000112a <+159>: mov %rax,%rdi |
| 0x000000000000112d <+162>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x0000000000001132 <+167>: lea 0x467(%rip),%rsi # 0x15a0 | Call another output stream and the operator << again. |
| 0x0000000000001139 <+174>: mov %rax,%rdi |
| 0x000000000000113c <+177>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x0000000000001141 <+182>: mov %rax,%rdx | Move option 2 (Sarah Davis) to eax then move to esi. . After moving rdx to rdi, call operator “<<” . |
| 0x0000000000001144 <+185>: mov 0x200f1a(%rip),%eax # 0x202064 <num2> |
| 0x000000000000114a <+191>: mov %eax,%esi |
| 0x000000000000114c <+193>: mov %rdx,%rdi |
| 0x000000000000114f <+196>: callq 0xd00 <\_ZNSolsEi@plt> |
| 0x0000000000001154 <+201>: mov %rax,%rdx | After moving registers, call operator “<<” for the output stream that will print outcome. |
| 0x0000000000001157 <+204>: mov 0x200e72(%rip),%rax # 0x201fd0 |
| 0x000000000000115e <+211>: mov %rax,%rsi |
| 0x0000000000001161 <+214>: mov %rdx,%rdi |
| 0x0000000000001164 <+217>: callq 0xca0 <\_ZNSolsEPFRSoS\_E@plt> |
| 0x0000000000001169 <+222>: lea 0x446(%rip),%rsi # 0x15b6 | Point to the address for “cout” and then call the “<<” operator to print the characters in cout. |
| 0x0000000000001170 <+229>: lea 0x200f29(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4> |
| 0x0000000000001177 <+236>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x000000000000117c <+241>: lea 0x200ead(%rip),%rsi # 0x202030 <name3> | Point name3 (Amy Friendly) to %rsi. Move %rax to %rdi. Call the output stream and the operator <<. |
| 0x0000000000001183 <+248>: mov %rax,%rdi |
| 0x0000000000001186 <+251>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x000000000000118b <+256>: lea 0x40e(%rip),%rsi # 0x15a0 | Call another output stream and the operator << again. |
| 0x0000000000001192 <+263>: mov %rax,%rdi |
| 0x0000000000001195 <+266>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x000000000000119a <+271>: mov %rax,%rdx | Move option 3 (Amy Friendly) to eax then to rsi. After moving rdx to rdi, call operator “<<” . |
| 0x000000000000119d <+274>: mov 0x200ec5(%rip),%eax # 0x202068 <num3> |
| 0x00000000000011a3 <+280>: mov %eax,%esi |
| 0x00000000000011a5 <+282>: mov %rdx,%rdi |
| 0x00000000000011a8 <+285>: callq 0xd00 <\_ZNSolsEi@plt> |
| 0x00000000000011ad <+290>: mov %rax,%rdx | After moving registers, call operator “<<” for the output stream that will print outcome. |
| 0x00000000000011b0 <+293>: mov 0x200e19(%rip),%rax # 0x201fd0 |
| 0x00000000000011b7 <+300>: mov %rax,%rsi |
| 0x00000000000011ba <+303>: mov %rdx,%rdi |
| 0x00000000000011bd <+306>: callq 0xca0 <\_ZNSolsEPFRSoS\_E@plt> |
| 0x00000000000011c2 <+311>: lea 0x3f1(%rip),%rsi # 0x15ba | Point to the address for “cout” and then call the “<<” operator to print the characters in cout. |
| 0x00000000000011c9 <+318>: lea 0x200ed0(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4> |
| 0x00000000000011d0 <+325>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x00000000000011d5 <+330>: lea 0x200e64(%rip),%rsi # 0x202040 <name4> | Point name4 (Johnny Smith) to %rsi. Move %rax to %rdi. Call the output stream and the operator <<. |
| 0x00000000000011dc <+337>: mov %rax,%rdi |
| 0x00000000000011df <+340>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x00000000000011e4 <+345>: lea 0x3b5(%rip),%rsi # 0x15a0 | Call another output stream and the operator << again. |
| 0x00000000000011eb <+352>: mov %rax,%rdi |
| 0x00000000000011ee <+355>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x00000000000011f3 <+360>: mov %rax,%rdx | Move option 4 (Johnny Smith) to eax then to rsi. After moving rdx to rdi, call operator “<<” . |
| 0x00000000000011f6 <+363>: mov 0x200e70(%rip),%eax # 0x20206c <num4> |
| 0x00000000000011fc <+369>: mov %eax,%esi |
| 0x00000000000011fe <+371>: mov %rdx,%rdi |
| 0x0000000000001201 <+374>: callq 0xd00 <\_ZNSolsEi@plt> |
| 0x0000000000001206 <+379>: mov %rax,%rdx | After moving registers, call operator “<<” for the output stream that will print outcome. |
| 0x0000000000001209 <+382>: mov 0x200dc0(%rip),%rax # 0x201fd0 |
| 0x0000000000001210 <+389>: mov %rax,%rsi |
| 0x0000000000001213 <+392>: mov %rdx,%rdi |
| 0x0000000000001216 <+395>: callq 0xca0 <\_ZNSolsEPFRSoS\_E@plt> |
| 0x000000000000121b <+400>: lea 0x39c(%rip),%rsi # 0x15be | Point to the address for “cout” and then call the “<<” operator to print the characters in cout. |
| 0x0000000000001222 <+407>: lea 0x200e77(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4> |
| 0x0000000000001229 <+414>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x000000000000122e <+419>: lea 0x200e1b(%rip),%rsi # 0x202050 <name5> | Point name5 (Carol Spears) to %rsi. Move %rax to %rdi. Call the output stream and the operator <<. |
| 0x0000000000001235 <+426>: mov %rax,%rdi |
| 0x0000000000001238 <+429>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x000000000000123d <+434>: lea 0x35c(%rip),%rsi # 0x15a0 | Call another output stream and the operator << again. |
| 0x0000000000001244 <+441>: mov %rax,%rdi |
| 0x0000000000001247 <+444>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt> |
| 0x000000000000124c <+449>: mov %rax,%rdx | Move option 5 (Carol Spears) to eax then to rsi. After moving rdx to rdi, call operator “<<” . |
| 0x000000000000124f <+452>: mov 0x200e1b(%rip),%eax # 0x202070 <num5> |
| 0x0000000000001255 <+458>: mov %eax,%esi |
| 0x0000000000001257 <+460>: mov %rdx,%rdi |
| 0x000000000000125a <+463>: callq 0xd00 <\_ZNSolsEi@plt> |
| 0x000000000000125f <+468>: mov %rax,%rdx | After moving registers, call operator “<<” for the output stream that will print outcome. |
| 0x0000000000001262 <+471>: mov 0x200d67(%rip),%rax # 0x201fd0 |
| 0x0000000000001269 <+478>: mov %rax,%rsi |
| 0x000000000000126c <+481>: mov %rdx,%rdi |
| 0x000000000000126f <+484>: callq 0xca0 <\_ZNSolsEPFRSoS\_E@plt> |
| 0x0000000000001274 <+489>: nop | Return to main menu |
| 0x0000000000001275 <+490>: pop %rbp |
| 0x0000000000001276 <+491>: retq |