# 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 0x0( rip), rsi  lea 0x0( rip), rdi  call 0x17 | Output “Hello! Welcome to our investment company” |
| call 0x1c <main+28>  mov eax,0x0( rip)  mov 0x0( rip), eax | answer equals result of CheckUserPermissionAccess |
| cmp 0x1, eax  je 0x40 <main+64> | If answer does not equal 1 jump to 0x40 |
| lea 0x0( rip), rsi  lea 0x0( rip), rdi  call 0x40 | Outputs "Invalid Password. Please try again..” |
| mov 0x0( rip), eax  cmp 0x1, eax  je 0x4d  jump 0x17 | do while loop that checks if answer equals 1, if it does jump to 0x4d, if it does not jump to 0x17 and start the process over |
| lea 0x0( rip), rsi  lea 0x0( rip), rdi  call 0x60  lea 0x0( rip), rsi  lea 0x0( rip), rdi  call 0x73  lea 0x0( rip), rsi  lea 0x0( rip), rdi  call 0x86  lea 0x0( rip), rsi  lea 0x0( rip), rdi  call 0x99 | Start of while loop  Output menu options:  "What would you like to do?\n"  "DISPLAY the client list (enter 1)”  "CHANGE a client's choice (enter 2)”  "Exit the program.. (enter 3)\n" |
| lea 0x0( rip), rsi  lea 0x0( rip),  call 0xac | Read user input and store into choice variable |
| lea 0x0( rip), rsi  lea 0x0( rip),  call 0xbf  mov rax, rdx  mov 0x0( rip), eax  mov eax, esi  mov rdx, rdi  call 0xd2  mov rax, rdx  mov 0x0( rip), rax  mov rax, rsi  mov rdx, rdi  call 0xe7 | Output “You chose “ plus the value of choice variable |
| mov 0x0( rip), eax  cmp 0x1, eax  jne 0xf9  mall DisplayInfo  jmp 0x109 | Switch statement that has a case for all options  For the first case if choice not equal to 1 continue to next case by jumping to 0xf9  If choice equals 1 call the DisplayInfo function and then jump to 0x109 |
| mov 0x0( rip), eax  cmp 0x2, eax  jne 0x109  call 0x109  jmp 0x109 | Second case, if choice not equal to 2 go to next case jump to 0x109  If choice equals 2 then call ChangeCustomerChoice function and then jump to 0x109 |
| mov 0x0( rip), eax  cmp 0x3, eax  je 0x119  jmp 0x4d | Do while choice does not equal 3 by looping back to 0x4d  Third case, if choice does not equal continue loop  If choice equals 3 then jump to 0x119 |
| mov 0x0, eax  pop rbp  ret | Exit and return 0 |

### ChangeCustomerChoice function

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  lea 0x0(%rip), %rsi  lea 0x0(%rip), %rdi  call 0x444 | Create variable changechoice  Output “Enter the number of the client that you wish to change.” |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x457 | Read user input and assign to variable changechoice |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x46a | Output “Please enter the client's new service choice (1 = Brokerage, 2 = Retirement)” |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x47d | Read user input and assign to variable newservice |
| mov 0x0(%rip),%eax  cmp $0x1,%eax  jne 0x496  mov 0x0(%rip),%eax  mov %eax,0x0(%rip)  jmp 0x4af | If changechoice does not equal 1 jump to 0x496 else changechoice does equal 1 then set value of variable num1 = newservice and then jump to 0x4af |
| mov 0x0(%rip),%eax  cmp $0x2,%eax  jne 0x4f8  mov 0x0(%rip),%eax  mov %eax,0x0(%rip)  jmp 0x4af | else if changechoice does not equal 2 jump to 0x4f8 else set value of variable num2 = newservice and then jump to 0x4af |
| mov 0x0(%rip),%eax  cmp $0x3,%eax  jne 0x4c8  mov 0x0(%rip),%eax  mov %eax,0x0(%rip)  jmp 0x4f8 | else if changechoice does not equal 3 jump to 0x4c8 else set value of variable num3 = newservice and then jump to 0x4f8 |
| mov 0x0(%rip),%eax  cmp $0x4,%eax  jne 0xe1  mov 0x0(%rip),%eax  mov %eax,0x0(%rip)  jmp 0x4f8 | else If changechoice does not equal 4 jump to 0x4e1 else changechoice does equal 4 then set value of variable num4 = newservice and then jump to 0x4f8 |
| mov 0x0(%rip),%eax  cmp $0x5,%eax  jne 0x4f8  mov 0x0(%rip),%eax  mov %eax,0x0(%rip) | else newservice = changechoice and if newservice does not equal 5 then jump to 0x4f8 and else num5 = newservice |
| nop  pop %rbp  ret | return value of newservice |

### 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  call 0x144  lea -0x45(%rbp),%rdx  lea -0x40(%rbp),%rax  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x15b  lea -0x45(%rbp),%rax  mov %rax,%rdi  call 0x167 | Declare two variables, username and password and then set their value to 0 and setting a lot of string characters to variables and then outputting the values of those variables |
| movl $0x0,-0x44(%rbp)  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x181 | Set the string value “Enter your username” to variable and then outputs variable. |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x194 | Read user input and store result into username variable |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x1a7 | Output string “Enter your password” |
| lea -0x40(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi  call 0x1ba | Read user input and store the result in the password variable as a string |
| lea -0x40(%rbp),%rax  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x1cd | Get value from answer variable |
| mov %eax,-0x44(%rbp)  cmpl $0x0,-0x44(%rbp)  jne 0x1dd  mov $0x1,%ebx  jmp 0x1e2  mov $0x2,%ebx | If answer does not equal zero then jump to 0x1dd and the result is 1 else the result is 2 |
| lea -0x40(%rbp),%rax  mov %rax,%rdi  call 0x1ee | Get the value of the answer variable |
| mov %ebx,%eax  mov -0x18(%rbp),%rcx  xor %fs:0x28,%rcx  je 0x23a | If value of result equals zero the jump to 0x23a else jump to 0x1ff |
| add $0x48,%rsp  pop %rbx  pop %rbp  ret | Add result to answer and then pop it from the stack and return |

### DisplayInfo Function

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x258  mov %rax,%rdx  mov 0x0(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x26d | Output a message “ Client's Name Service Selected (1 = Brokerage, 2 = Retirement)” and start a new line |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x280  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x28f  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x29e  mov %rax,%rdx  mov 0x0(%rip),%eax  mov %eax,%esi  mov %rdx,%rdi  call 0x2b1  mov %rax,%rdx  mov 0x0(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x2c6 | Output “std:cout << “1. “ << “Bob Jones” << “selected option” << num1 << endl  This outputs the following “1. Bob Jones selected option 1” and ends with a new line |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x2d9  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x2e8  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x2f7  mov %rax,%rdx  mov 0x0(%rip),%eax  mov %eax,%esi  mov %rdx,%rdi  call 0x30a  mov %rax,%rdx  mov 0x0(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x31f | Output “std:cout << “2. “ << “Sarah Davis” << “selected option” << 1 << endl  This outputs the following “2. Sarah Davis selected option 2” and ends with a new line |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x332  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x341  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x350  mov %rax,%rdx  mov 0x0(%rip),%eax  mov %eax,%esi  mov %rdx,%rdi  call 0x363  mov %rax,%rdx  mov 0x0(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x378 | Output “std:cout << “3. “ << “Amy Friendly” << “selected option” << 3 << endl  This outputs the following “3. Amy Friendly selected option 3” and ends with a new line |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x38b  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x39a  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x3a9  mov %rax,%rdx  mov 0x0(%rip),%eax  mov %eax,%esi  mov %rdx,%rdi  call 0x3bc  mov %rax,%rdx  mov 0x0(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x3d1 | Output “std:cout << “4. “ << “Johnny Smith” << “selected option” << 4 << endl  This outputs the following “4. Johnny Smith selected option 4” and ends with a new line |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x3e4  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x3f3  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x402  mov %rax,%rdx  mov 0x0(%rip),%eax  mov %eax,%esi  mov %rdx,%rdi  call 0x415  mov %rax,%rdx  mov 0x0(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x42a | Output “std:cout << “5. “ << “Carol Spears” << “selected option” << 5 << endl  This outputs the following “5. Carol Spears selected option 5” and ends with a new line |
| nop  pop %rbp  ret | return 1 |