**6-2 Activity: Binary to C++ With Security Vulnerabilities**

Dylan Jeffery

SNHU

CS-410

Brian West

2/13/2022

# **CS 410 Binary to C++ With Security Vulnerabilities Activity Template**

**Step 1:** Convert the binary file to assembly code.

**0000000000000000 <\_Z11DisplayMenuv>:**

**push %rbp**

**mov %rsp,%rbp**

**lea 0x0(%rip),%rsi # b <\_Z11DisplayMenuv+0xb>**

**lea 0x0(%rip),%rdi # 12 <\_Z11DisplayMenuv+0x12>**

**callq 17 <\_Z11DisplayMenuv+0x17>**

**lea 0x0(%rip),%rsi # 1e <\_Z11DisplayMenuv+0x1e>**

**lea 0x0(%rip),%rdi # 25 <\_Z11DisplayMenuv+0x25>**

**callq 2a <\_Z11DisplayMenuv+0x2a>**

**lea 0x0(%rip),%rsi # 31 <\_Z11DisplayMenuv+0x31>**

**lea 0x0(%rip),%rdi # 38 <\_Z11DisplayMenuv+0x38>**

**callq 3d <\_Z11DisplayMenuv+0x3d>**

**lea 0x0(%rip),%rsi # 44 <\_Z11DisplayMenuv+0x44>**

**lea 0x0(%rip),%rdi # 4b <\_Z11DisplayMenuv+0x4b>**

**callq 50 <\_Z11DisplayMenuv+0x50>**

**lea 0x0(%rip),%rsi # 57 <\_Z11DisplayMenuv+0x57>**

**lea 0x0(%rip),%rdi # 5e <\_Z11DisplayMenuv+0x5e>**

**callq 63 <\_Z11DisplayMenuv+0x63>**

**lea 0x0(%rip),%rsi # 6a <\_Z11DisplayMenuv+0x6a>**

**lea 0x0(%rip),%rdi # 71 <\_Z11DisplayMenuv+0x71>**

**callq 76 <\_Z11DisplayMenuv+0x76>**

**nop**

**pop %rbp**

**retq**

**0000000000000079 <main>:**

**push %rbp**

**mov %rsp,%rbp**

**sub $0x20,%rsp**

**mov %fs:0x28,%rax**

**mov %rax,-0x8(%rbp)**

**xor %eax,%eax**

**movl $0x0,-0x14(%rbp)**

**mov -0x14(%rbp),%eax**

**cmp $0x5,%eax**

**je 308 <main+0x28f>**

**lea 0x0(%rip),%rsi # aa <main+0x31>**

**lea 0x0(%rip),%rdi # b1 <main+0x38>**

**callq b6 <main+0x3d>**

**lea 0x0(%rip),%rsi # bd <main+0x44>**

**lea 0x0(%rip),%rdi # c4 <main+0x4b>**

**callq c9 <main+0x50>**

**lea 0x0(%rip),%rsi # d0 <main+0x57>**

**lea 0x0(%rip),%rdi # d7 <main+0x5e>**

**callq dc <main+0x63>**

**lea 0x0(%rip),%rsi # e3 <main+0x6a>**

**lea 0x0(%rip),%rdi # ea <main+0x71>**

**callq ef <main+0x76>**

**lea 0x0(%rip),%rsi # f6 <main+0x7d>**

**lea 0x0(%rip),%rdi # fd <main+0x84>**

**callq 102 <main+0x89>**

**lea 0x0(%rip),%rsi # 109 <main+0x90>**

**lea 0x0(%rip),%rdi # 110 <main+0x97>**

**callq 115 <main+0x9c>**

**lea -0x14(%rbp),%rax**

**mov %rax,%rsi**

**lea 0x0(%rip),%rdi # 123 <main+0xaa>**

**callq 128 <main+0xaf>**

**mov -0x14(%rbp),%eax**

**cmp $0x1,%eax**

**jne 1c9 <main+0x150>**

**lea -0x10(%rbp),%rax**

**mov %rax,%rsi**

**lea 0x0(%rip),%rdi # 142 <main+0xc9>**

**callq 147 <main+0xce>**

**mov %rax,%rdx**

**lea -0xc(%rbp),%rax**

**mov %rax,%rsi**

**mov %rdx,%rdi**

**callq 159 <main+0xe0>**

**mov -0x10(%rbp),%eax**

**mov %eax,%esi**

**lea 0x0(%rip),%rdi # 165 <main+0xec>**

**callq 16a <main+0xf1>**

**lea 0x0(%rip),%rsi # 171 <main+0xf8>**

**mov %rax,%rdi**

**callq 179 <main+0x100>**

**mov %rax,%rdx**

**mov -0xc(%rbp),%eax**

**mov %eax,%esi**

**mov %rdx,%rdi**

**callq 189 <main+0x110>**

**lea 0x0(%rip),%rsi # 190 <main+0x117>**

**mov %rax,%rdi**

**callq 198 <main+0x11f>**

**mov %rax,%rcx**

**mov -0x10(%rbp),%edx**

**mov -0xc(%rbp),%eax**

**sub %eax,%edx**

**mov %edx,%eax**

**mov %eax,%esi**

**mov %rcx,%rdi**

**callq 1af <main+0x136>**

**mov %rax,%rdx**

**mov 0x0(%rip),%rax # 1b9 <main+0x140>**

**mov %rax,%rsi**

**mov %rdx,%rdi**

**callq 1c4 <main+0x14b>**

**jmpq 97 <main+0x1e>**

**mov -0x14(%rbp),%eax**

**cmp $0x2,%eax**

**jne 268 <main+0x1ef>**

**lea -0x10(%rbp),%rax**

**mov %rax,%rsi**

**lea 0x0(%rip),%rdi # 1e3 <main+0x16a>**

**callq 1e8 <main+0x16f>**

**mov %rax,%rdx**

**lea -0xc(%rbp),%rax**

**mov %rax,%rsi**

**mov %rdx,%rdi**

**callq 1fa <main+0x181>**

**mov -0x10(%rbp),%eax**

**mov %eax,%esi**

**lea 0x0(%rip),%rdi # 206 <main+0x18d>**

**callq 20b <main+0x192>**

**lea 0x0(%rip),%rsi # 212 <main+0x199>**

**mov %rax,%rdi**

**callq 21a <main+0x1a1>**

**mov %rax,%rdx**

**mov -0xc(%rbp),%eax**

**mov %eax,%esi**

**mov %rdx,%rdi**

**callq 22a <main+0x1b1>**

**lea 0x0(%rip),%rsi # 231 <main+0x1b8>**

**mov %rax,%rdi**

**callq 239 <main+0x1c0>**

**mov %rax,%rcx**

**mov -0x10(%rbp),%edx**

**mov -0xc(%rbp),%eax**

**add %edx,%eax**

**mov %eax,%esi**

**mov %rcx,%rdi**

**callq 24e <main+0x1d5>**

**mov %rax,%rdx**

**mov 0x0(%rip),%rax # 258 <main+0x1df>**

**mov %rax,%rsi**

**mov %rdx,%rdi**

**callq 263 <main+0x1ea>**

**jmpq 97 <main+0x1e>**

**mov -0x14(%rbp),%eax**

**cmp $0x3,%eax**

**jne 97 <main+0x1e>**

**lea -0x10(%rbp),%rax**

**mov %rax,%rsi**

**lea 0x0(%rip),%rdi # 282 <main+0x209>**

**callq 287 <main+0x20e>**

**mov %rax,%rdx**

**lea -0xc(%rbp),%rax**

**mov %rax,%rsi**

**mov %rdx,%rdi**

**callq 299 <main+0x220>**

**mov -0x10(%rbp),%eax**

**mov %eax,%esi**

**lea 0x0(%rip),%rdi # 2a5 <main+0x22c>**

**callq 2aa <main+0x231>**

**lea 0x0(%rip),%rsi # 2b1 <main+0x238>**

**mov %rax,%rdi**

**callq 2b9 <main+0x240>**

**mov %rax,%rdx**

**mov -0xc(%rbp),%eax**

**mov %eax,%esi**

**mov %rdx,%rdi**

**callq 2c9 <main+0x250>**

**lea 0x0(%rip),%rsi # 2d0 <main+0x257>**

**mov %rax,%rdi**

**callq 2d8 <main+0x25f>**

**mov %rax,%rcx**

**mov -0x10(%rbp),%eax**

**mov -0xc(%rbp),%esi**

**cltd**

**idiv %esi**

**mov %eax,%esi**

**mov %rcx,%rdi**

**callq 2ee <main+0x275>**

**mov %rax,%rdx**

**mov 0x0(%rip),%rax # 2f8 <main+0x27f>**

**mov %rax,%rsi**

**mov %rdx,%rdi**

**callq 303 <main+0x28a>**

**jmpq 97 <main+0x1e>**

**mov $0x0,%eax**

**mov -0x8(%rbp),%rcx**

**xor %fs:0x28,%rcx**

**je 321 <main+0x2a8>**

**callq 321 <main+0x2a8>**

**leaveq**

**retq**

**Step 2:** Explain the functionality of the blocks of assembly code.

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  lea 0x0(%rip),%rsi # b <\_Z11DisplayMenuv+0xb>  lea 0x0(%rip),%rdi # 12 <\_Z11DisplayMenuv+0x12>  callq 17 <\_Z11DisplayMenuv+0x17>  lea 0x0(%rip),%rsi # 1e <\_Z11DisplayMenuv+0x1e>  lea 0x0(%rip),%rdi # 25 <\_Z11DisplayMenuv+0x25>  callq 2a <\_Z11DisplayMenuv+0x2a>  lea 0x0(%rip),%rsi # 31 <\_Z11DisplayMenuv+0x31>  lea 0x0(%rip),%rdi # 38 <\_Z11DisplayMenuv+0x38>  callq 3d <\_Z11DisplayMenuv+0x3d>  lea 0x0(%rip),%rsi # 44 <\_Z11DisplayMenuv+0x44>  lea 0x0(%rip),%rdi # 4b <\_Z11DisplayMenuv+0x4b>  callq 50 <\_Z11DisplayMenuv+0x50>  lea 0x0(%rip),%rsi # 57 <\_Z11DisplayMenuv+0x57>  lea 0x0(%rip),%rdi # 5e <\_Z11DisplayMenuv+0x5e>  callq 63 <\_Z11DisplayMenuv+0x63>  lea 0x0(%rip),%rsi # 6a <\_Z11DisplayMenuv+0x6a>  lea 0x0(%rip),%rdi # 71 <\_Z11DisplayMenuv+0x71>  callq 76 <\_Z11DisplayMenuv+0x76>  nop  pop %rbp  retq | Initialize some variables and print welcome statements |
| push %rbp  mov %rsp,%rbp  sub $0x20,%rsp  mov %fs:0x28,%rax  mov %rax,-0x8(%rbp)  xor %eax,%eax  movl $0x0,-0x14(%rbp)  mov -0x14(%rbp),%eax  cmp $0x5,%eax  je 308 <main+0x28f>  lea 0x0(%rip),%rsi # aa <main+0x31>  lea 0x0(%rip),%rdi # b1 <main+0x38>  callq b6 <main+0x3d>  lea 0x0(%rip),%rsi # bd <main+0x44>  lea 0x0(%rip),%rdi # c4 <main+0x4b>  callq c9 <main+0x50>  lea 0x0(%rip),%rsi # d0 <main+0x57>  lea 0x0(%rip),%rdi # d7 <main+0x5e>  callq dc <main+0x63>  lea 0x0(%rip),%rsi # e3 <main+0x6a>  lea 0x0(%rip),%rdi # ea <main+0x71>  callq ef <main+0x76>  lea 0x0(%rip),%rsi # f6 <main+0x7d>  lea 0x0(%rip),%rdi # fd <main+0x84>  callq 102 <main+0x89>  lea 0x0(%rip),%rsi # 109 <main+0x90>  lea 0x0(%rip),%rdi # 110 <main+0x97>  callq 115 <main+0x9c>  lea -0x14(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi # 123 <main+0xaa>  callq 128 <main+0xaf>  mov -0x14(%rbp),%eax | More variables, and a comparison for equality before load effective address’s for variables and calls (likely cout or cin) en masse. |
| cmp $0x1,%eax  jne 1c9 <main+0x150>  lea -0x10(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi # 142 <main+0xc9>  callq 147 <main+0xce>  mov %rax,%rdx  lea -0xc(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 159 <main+0xe0>  mov -0x10(%rbp),%eax  mov %eax,%esi  lea 0x0(%rip),%rdi # 165 <main+0xec>  callq 16a <main+0xf1>  lea 0x0(%rip),%rsi # 171 <main+0xf8>  mov %rax,%rdi  callq 179 <main+0x100>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 189 <main+0x110>  lea 0x0(%rip),%rsi # 190 <main+0x117>  mov %rax,%rdi  callq 198 <main+0x11f>  mov %rax,%rcx  mov -0x10(%rbp),%edx  mov -0xc(%rbp),%eax  sub %eax,%edx  mov %edx,%eax  mov %eax,%esi  mov %rcx,%rdi  callq 1af <main+0x136>  mov %rax,%rdx  mov 0x0(%rip),%rax # 1b9 <main+0x140>  mov %rax,%rsi  mov %rdx,%rdi  callq 1c4 <main+0x14b> | Start of the menu loop, provide the user with their options |
| jmpq 97 <main+0x1e>  mov -0x14(%rbp),%eax  cmp $0x2,%eax  jne 268 <main+0x1ef>  lea -0x10(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi # 1e3 <main+0x16a>  callq 1e8 <main+0x16f>  mov %rax,%rdx  lea -0xc(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 1fa <main+0x181>  mov -0x10(%rbp),%eax  mov %eax,%esi  lea 0x0(%rip),%rdi # 206 <main+0x18d>  callq 20b <main+0x192>  lea 0x0(%rip),%rsi # 212 <main+0x199>  mov %rax,%rdi  callq 21a <main+0x1a1>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 22a <main+0x1b1>  lea 0x0(%rip),%rsi # 231 <main+0x1b8>  mov %rax,%rdi  callq 239 <main+0x1c0>  mov %rax,%rcx  mov -0x10(%rbp),%edx  mov -0xc(%rbp),%eax  add %edx,%eax  mov %eax,%esi  mov %rcx,%rdi  callq 24e <main+0x1d5>  mov %rax,%rdx  mov 0x0(%rip),%rax # 258 <main+0x1df>  mov %rax,%rsi  mov %rdx,%rdi  callq 263 <main+0x1ea> | Conditional logic determines the mathematical operation selected, takes x & y variables, and prints out the results formatted |
| jmpq 97 <main+0x1e>  mov -0x14(%rbp),%eax  cmp $0x3,%eax  jne 97 <main+0x1e>  lea -0x10(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi # 282 <main+0x209>  callq 287 <main+0x20e>  mov %rax,%rdx  lea -0xc(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 299 <main+0x220>  mov -0x10(%rbp),%eax  mov %eax,%esi  lea 0x0(%rip),%rdi # 2a5 <main+0x22c>  callq 2aa <main+0x231>  lea 0x0(%rip),%rsi # 2b1 <main+0x238>  mov %rax,%rdi  callq 2b9 <main+0x240>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 2c9 <main+0x250>  lea 0x0(%rip),%rsi # 2d0 <main+0x257>  mov %rax,%rdi  callq 2d8 <main+0x25f>  mov %rax,%rcx  mov -0x10(%rbp),%eax  mov -0xc(%rbp),%esi  cltd  idiv %esi  mov %eax,%esi  mov %rcx,%rdi  callq 2ee <main+0x275>  mov %rax,%rdx  mov 0x0(%rip),%rax # 2f8 <main+0x27f>  mov %rax,%rsi  mov %rdx,%rdi  callq 303 <main+0x28a>  jmpq 97 <main+0x1e>  mov $0x0,%eax  mov -0x8(%rbp),%rcx  xor %fs:0x28,%rcx  je 321 <main+0x2a8>  callq 321 <main+0x2a8>  leaveq  retq | Conditional logic determines the mathematical operation selected, takes x & y variables, and prints out the results formatted  Then show the menu again |

**Step 3:** Convert the C++ code to binary.

**7f45 4c46 0201 0100 0000 0000 0000 0000**

**0100 3e00 0100 0000 0000 0000 0000 0000**

**0000 0000 0000 0000 b814 0000 0000 0000**

**0000 0000 4000 0000 0000 4000 0f00 0e00**

**5548 89e5 4883 ec20 6448 8b04 2528 0000**

**0048 8945 f831 c0c7 45f0 0100 0000 488d**

**3500 0000 0048 8d3d 0000 0000 e800 0000**

**0048 89c2 488b 0500 0000 0048 89c6 4889**

**d7e8 0000 0000 488d 3500 0000 0048 8d3d**

**0000 0000 e800 0000 0048 89c2 488b 0500**

**0000 0048 89c6 4889 d7e8 0000 0000 488d**

**3500 0000 0048 8d3d 0000 0000 e800 0000**

**0048 89c2 488b 0500 0000 0048 89c6 4889**

**d7e8 0000 0000 488d 3500 0000 0048 8d3d**

**0000 0000 e800 0000 0048 89c2 488b 0500**

**0000 0048 89c6 4889 d7e8 0000 0000 488d**

**3500 0000 0048 8d3d 0000 0000 e800 0000**

**0048 89c2 488b 0500 0000 0048 89c6 4889**

**d7e8 0000 0000 488d 3500 0000 0048 8d3d**

**0000 0000 e800 0000 0048 89c2 488b 0500**

**0000 0048 89c6 4889 d7e8 0000 0000 488d**

**45ec 4889 c648 8d3d 0000 0000 e800 0000**

**0048 8d3d 0000 0000 e800 0000 0084 c074**

**62be 0000 0000 488d 3d00 0000 00e8 0000**

**0000 ba0a 0000 00be ffff ff7f 488d 3d00**

**0000 00e8 0000 0000 488d 3500 0000 0048**

**8d3d 0000 0000 e800 0000 0048 8d35 0000**

**0000 488d 3d00 0000 00e8 0000 0000 488d**

**45ec 4889 c648 8d3d 0000 0000 e800 0000**

**00eb 8e83 7df0 000f 8424 0200 008b 45ec**

**83f8 010f 85a5 0000 0048 8d45 e448 89c6**

**488d 3d00 0000 00e8 0000 0000 4889 c248**

**8d45 e848 89c6 4889 d7e8 0000 0000 8b55**

**e48b 45e8 01d0 8945 f48b 45e4 89c6 488d**

**3d00 0000 00e8 0000 0000 488d 3500 0000**

**0048 89c7 e800 0000 0048 89c2 8b45 e889**

**c648 89d7 e800 0000 0048 8d35 0000 0000**

**4889 c7e8 0000 0000 4889 c28b 45f4 89c6**

**4889 d7e8 0000 0000 4889 c248 8b05 0000**

**0000 4889 c648 89d7 e800 0000 00c7 45f0**

**0000 0000 e800 0000 00e9 45ff ffff 8b45**

**ec83 f802 0f85 9200 0000 488d 45e4 4889**

**c648 8d3d 0000 0000 e800 0000 0048 89c2**

**488d 45e8 4889 c648 89d7 e800 0000 008b**

**55e4 8b45 e829 c289 d089 45f4 8b45 e489**

**c648 8d3d 0000 0000 e800 0000 0048 8d35**

**0000 0000 4889 c7e8 0000 0000 4889 c28b**

**45e8 89c6 4889 d7e8 0000 0000 488d 3500**

**0000 0048 89c7 e800 0000 0048 89c2 8b45**

**f489 c648 89d7 e800 0000 00c7 45f0 0000**

**0000 e800 0000 00e9 a7fe ffff 8b45 ec83**

**f803 0f85 9100 0000 488d 45e4 4889 c648**

**8d3d 0000 0000 e800 0000 0048 89c2 488d**

**45e8 4889 c648 89d7 e800 0000 008b 55e4**

**8b45 e80f afc2 8945 f48b 45e4 89c6 488d**

**3d00 0000 00e8 0000 0000 488d 3500 0000**

**0048 89c7 e800 0000 0048 89c2 8b45 e889**

**c648 89d7 e800 0000 0048 8d35 0000 0000**

**4889 c7e8 0000 0000 4889 c28b 45f4 89c6**

**4889 d7e8 0000 0000 c745 f000 0000 00e8**

**0000 0000 e90a feff ff8b 45ec 83f8 037e**

**0cc7 45f0 0000 0000 e9f6 fdff ff8b 45ec**

**85c0 7f11 c745 f000 0000 00e8 0000 0000**

**e9de fdff ffc7 45f0 0100 0000 e9d2 fdff**

**ff90 488b 45f8 6448 3304 2528 0000 0074**

**05e8 0000 0000 c9c3 5548 89e5 e800 0000**

**00b8 0000 0000 5dc3 5548 89e5 4883 ec10**

**897d fc89 75f8 837d fc01 7532 817d f8ff**

**ff00 0075 2948 8d3d 0000 0000 e800 0000**

**0048 8d15 0000 0000 488d 3500 0000 0048**

**8b05 0000 0000 4889 c7e8 0000 0000 90c9**

**c355 4889 e5be ffff 0000 bf01 0000 00e8**

**a4ff ffff 5dc3 002d 2d2d 2d2d 2d2d 2d2d**

**2d2d 2d2d 2d2d 2d00 2d20 3129 4164 6420**

**2d00 2d20 3229 5375 6274 7261 6374 202d**

**002d 2033 294d 756c 7469 706c 7920 2d00**

**2d20 3429 4578 6974 202d 0059 6f75 2063**

**616e 206f 6e6c 7920 656e 7465 7220 6e75**

**6d62 6572 732e 0a00 456e 7465 7220 6120**

**6e75 6d62 6572 2e0a 0020 2b20 0020 3d20**

**0020 2d20 0020 2a20 0000 0000 0000 0000**

**0000 0000 0000 0000 0047 4343 3a20 2855**

**6275 6e74 7520 372e 352e 302d 3375 6275**

**6e74 7531 7e31 382e 3034 2920 372e 352e**

**3000 0000 0000 0000 1400 0000 0000 0000**

**017a 5200 0178 1001 1b0c 0708 9001 0000**

**1c00 0000 1c00 0000 0000 0000 d803 0000**

**0041 0e10 8602 430d 0603 d303 0c07 0800**

**1c00 0000 3c00 0000 0000 0000 1000 0000**

**0041 0e10 8602 430d 064b 0c07 0800 0000**

**1c00 0000 5c00 0000 0000 0000 4900 0000**

**0041 0e10 8602 430d 0602 440c 0708 0000**

**1c00 0000 7c00 0000 0000 0000 1500 0000**

**0041 0e10 8602 430d 0650 0c07 0800 0000**

**0000 0000 0000 0000 0000 0000 0000 0000**

**0000 0000 0000 0000 0100 0000 0400 f1ff**

**0000 0000 0000 0000 0000 0000 0000 0000**

**0000 0000 0300 0100 0000 0000 0000 0000**

**0000 0000 0000 0000 0000 0000 0300 0300**

**0000 0000 0000 0000 0000 0000 0000 0000**

**0000 0000 0300 0400 0000 0000 0000 0000**

**0000 0000 0000 0000 0000 0000 0300 0500**

**0000 0000 0000 0000 0000 0000 0000 0000**

**1a00 0000 0100 0500 0000 0000 0000 0000**

**0100 0000 0000 0000 3500 0000 0100 0400**

**0000 0000 0000 0000 0100 0000 0000 0000**

**4400 0000 0200 0100 e803 0000 0000 0000**

**4900 0000 0000 0000 7400 0000 0200 0100**

**3104 0000 0000 0000 1500 0000 0000 0000**

**0000 0000 0300 0600 0000 0000 0000 0000**

**0000 0000 0000 0000 0000 0000 0300 0900**

**0000 0000 0000 0000 0000 0000 0000 0000**

**0000 0000 0300 0a00 0000 0000 0000 0000**

**0000 0000 0000 0000 0000 0000 0300 0800**

**0000 0000 0000 0000 0000 0000 0000 0000**

**8300 0000 1200 0100 0000 0000 0000 0000**

**d803 0000 0000 0000 9400 0000 1000 0000**

**0000 0000 0000 0000 0000 0000 0000 0000**

**9e00 0000 1000 0000 0000 0000 0000 0000**

**0000 0000 0000 0000 b400 0000 1000 0000**

**0000 0000 0000 0000 0000 0000 0000 0000**

**ec00 0000 1000 0000 0000 0000 0000 0000**

**0000 0000 0000 0000 2701 0000 1000 0000**

**0000 0000 0000 0000 0000 0000 0000 0000**

**3801 0000 1000 0000 0000 0000 0000 0000**

**0000 0000 0000 0000 4101 0000 1000 0000**

**0000 0000 0000 0000 0000 0000 0000 0000**

**4c01 0000 1000 0000 0000 0000 0000 0000**

**0000 0000 0000 0000 7901 0000 1000 0000**

**0000 0000 0000 0000 0000 0000 0000 0000**

**b501 0000 1000 0000 0000 0000 0000 0000**

**0000 0000 0000 0000 c501 0000 1000 0000**

**0000 0000 0000 0000 0000 0000 0000 0000**

**cf01 0000 1000 0000 0000 0000 0000 0000**

**0000 0000 0000 0000 e001 0000 1200 0100**

**d803 0000 0000 0000 1000 0000 0000 0000**

**e501 0000 1000 0000 0000 0000 0000 0000**

**0000 0000 0000 0000 fd01 0000 1002 0000**

**0000 0000 0000 0000 0000 0000 0000 0000**

**0a02 0000 1000 0000 0000 0000 0000 0000**

**0000 0000 0000 0000 2202 0000 1000 0000**

**0000 0000 0000 0000 0000 0000 0000 0000**

**0056 756c 6e65 7261 626c 6543 616c 6375**

**6c61 746f 722e 6370 7000 5f5a 5374 4c31**

**3970 6965 6365 7769 7365 5f63 6f6e 7374**

**7275 6374 005f 5a53 744c 385f 5f69 6f69**

**6e69 7400 5f5a 3431 5f5f 7374 6174 6963**

**5f69 6e69 7469 616c 697a 6174 696f 6e5f**

**616e 645f 6465 7374 7275 6374 696f 6e5f**

**3069 6900 5f47 4c4f 4241 4c5f 5f73 7562**

**5f49 5f5f 5a31 3164 6973 706c 6179 4d65**

**6e75 7600 5f5a 5374 3463 6f75 7400 5f47**

**4c4f 4241 4c5f 4f46 4653 4554 5f54 4142**

**4c45 5f00 5f5a 5374 6c73 4953 7431 3163**

**6861 725f 7472 6169 7473 4963 4545 5253**

**7431 3362 6173 6963 5f6f 7374 7265 616d**

**4963 545f 4553 355f 504b 6300 5f5a 5374**

**3465 6e64 6c49 6353 7431 3163 6861 725f**

**7472 6169 7473 4963 4545 5253 7431 3362**

**6173 6963 5f6f 7374 7265 616d 4954 5f54**

**305f 4553 365f 005f 5a4e 536f 6c73 4550**

**4652 536f 535f 4500 5f5a 5374 3363 696e**

**005f 5a4e 5369 7273 4552 6900 5f5a 4e4b**

**5374 3962 6173 6963 5f69 6f73 4963 5374**

**3131 6368 6172 5f74 7261 6974 7349 6345**

**4534 6661 696c 4576 005f 5a4e 5374 3962**

**6173 6963 5f69 6f73 4963 5374 3131 6368**

**6172 5f74 7261 6974 7349 6345 4535 636c**

**6561 7245 5374 3132 5f49 6f73 5f49 6f73**

**7461 7465 005f 5a4e 5369 3669 676e 6f72**

**6545 6c69 005f 5a4e 536f 6c73 4569 005f**

**5f73 7461 636b 5f63 686b 5f66 6169 6c00**

**6d61 696e 005f 5a4e 5374 3869 6f73 5f62**

**6173 6534 496e 6974 4331 4576 005f 5f64**

**736f 5f68 616e 646c 6500 5f5a 4e53 7438**

**696f 735f 6261 7365 3449 6e69 7444 3145**

**7600 5f5f 6378 615f 6174 6578 6974 0000**

**2100 0000 0000 0000 0200 0000 0500 0000**

**fdff ffff ffff ffff 2800 0000 0000 0000**

**0200 0000 0f00 0000 fcff ffff ffff ffff**

**2d00 0000 0000 0000 0400 0000 1100 0000**

**fcff ffff ffff ffff 3700 0000 0000 0000**

**2a00 0000 1200 0000 fcff ffff ffff ffff**

**4200 0000 0000 0000 0400 0000 1300 0000**

**fcff ffff ffff ffff 4900 0000 0000 0000**

**0200 0000 0500 0000 0e00 0000 0000 0000**

**5000 0000 0000 0000 0200 0000 0f00 0000**

**fcff ffff ffff ffff 5500 0000 0000 0000**

**0400 0000 1100 0000 fcff ffff ffff ffff**

**5f00 0000 0000 0000 2a00 0000 1200 0000**

**fcff ffff ffff ffff 6a00 0000 0000 0000**

**0400 0000 1300 0000 fcff ffff ffff ffff**

**7100 0000 0000 0000 0200 0000 0500 0000**

**1800 0000 0000 0000 7800 0000 0000 0000**

**0200 0000 0f00 0000 fcff ffff ffff ffff**

**7d00 0000 0000 0000 0400 0000 1100 0000**

**fcff ffff ffff ffff 8700 0000 0000 0000**

**2a00 0000 1200 0000 fcff ffff ffff ffff**

**9200 0000 0000 0000 0400 0000 1300 0000**

**fcff ffff ffff ffff 9900 0000 0000 0000**

**0200 0000 0500 0000 2700 0000 0000 0000**

**a000 0000 0000 0000 0200 0000 0f00 0000**

**fcff ffff ffff ffff a500 0000 0000 0000**

**0400 0000 1100 0000 fcff ffff ffff ffff**

**af00 0000 0000 0000 2a00 0000 1200 0000**

**fcff ffff ffff ffff ba00 0000 0000 0000**

**0400 0000 1300 0000 fcff ffff ffff ffff**

**c100 0000 0000 0000 0200 0000 0500 0000**

**3600 0000 0000 0000 c800 0000 0000 0000**

**0200 0000 0f00 0000 fcff ffff ffff ffff**

**cd00 0000 0000 0000 0400 0000 1100 0000**

**fcff ffff ffff ffff d700 0000 0000 0000**

**2a00 0000 1200 0000 fcff ffff ffff ffff**

**e200 0000 0000 0000 0400 0000 1300 0000**

**fcff ffff ffff ffff e900 0000 0000 0000**

**0200 0000 0500 0000 fdff ffff ffff ffff**

**f000 0000 0000 0000 0200 0000 0f00 0000**

**fcff ffff ffff ffff f500 0000 0000 0000**

**0400 0000 1100 0000 fcff ffff ffff ffff**

**ff00 0000 0000 0000 2a00 0000 1200 0000**

**fcff ffff ffff ffff 0a01 0000 0000 0000**

**0400 0000 1300 0000 fcff ffff ffff ffff**

**1801 0000 0000 0000 0200 0000 1400 0000**

**fcff ffff ffff ffff 1d01 0000 0000 0000**

**0400 0000 1500 0000 fcff ffff ffff ffff**

**2401 0000 0000 0000 0200 0000 1400 0000**

**0c00 0000 0000 0000 2901 0000 0000 0000**

**0400 0000 1600 0000 fcff ffff ffff ffff**

**3901 0000 0000 0000 0200 0000 1400 0000**

**0c00 0000 0000 0000 3e01 0000 0000 0000**

**0400 0000 1700 0000 fcff ffff ffff ffff**

**4f01 0000 0000 0000 0200 0000 1400 0000**

**fcff ffff ffff ffff 5401 0000 0000 0000**

**0400 0000 1800 0000 fcff ffff ffff ffff**

**5b01 0000 0000 0000 0200 0000 0500 0000**

**4100 0000 0000 0000 6201 0000 0000 0000**

**0200 0000 0f00 0000 fcff ffff ffff ffff**

**6701 0000 0000 0000 0400 0000 1100 0000**

**fcff ffff ffff ffff 6e01 0000 0000 0000**

**0200 0000 0500 0000 5e00 0000 0000 0000**

**7501 0000 0000 0000 0200 0000 0f00 0000**

**fcff ffff ffff ffff 7a01 0000 0000 0000**

**0400 0000 1100 0000 fcff ffff ffff ffff**

**8801 0000 0000 0000 0200 0000 1400 0000**

**fcff ffff ffff ffff 8d01 0000 0000 0000**

**0400 0000 1500 0000 fcff ffff ffff ffff**

**b301 0000 0000 0000 0200 0000 1400 0000**

**fcff ffff ffff ffff b801 0000 0000 0000**

**0400 0000 1500 0000 fcff ffff ffff ffff**

**ca01 0000 0000 0000 0400 0000 1500 0000**

**fcff ffff ffff ffff e101 0000 0000 0000**

**0200 0000 0f00 0000 fcff ffff ffff ffff**

**e601 0000 0000 0000 0400 0000 1900 0000**

**fcff ffff ffff ffff ed01 0000 0000 0000**

**0200 0000 0500 0000 6f00 0000 0000 0000**

**f501 0000 0000 0000 0400 0000 1100 0000**

**fcff ffff ffff ffff 0502 0000 0000 0000**

**0400 0000 1900 0000 fcff ffff ffff ffff**

**0c02 0000 0000 0000 0200 0000 0500 0000**

**7300 0000 0000 0000 1402 0000 0000 0000**

**0400 0000 1100 0000 fcff ffff ffff ffff**

**2402 0000 0000 0000 0400 0000 1900 0000**

**fcff ffff ffff ffff 2e02 0000 0000 0000**

**2a00 0000 1200 0000 fcff ffff ffff ffff**

**3902 0000 0000 0000 0400 0000 1300 0000**

**fcff ffff ffff ffff 4502 0000 0000 0000**

**0200 0000 0e00 0000 fcff ffff ffff ffff**

**6402 0000 0000 0000 0200 0000 1400 0000**

**fcff ffff ffff ffff 6902 0000 0000 0000**

**0400 0000 1500 0000 fcff ffff ffff ffff**

**7b02 0000 0000 0000 0400 0000 1500 0000**

**fcff ffff ffff ffff 9402 0000 0000 0000**

**0200 0000 0f00 0000 fcff ffff ffff ffff**

**9902 0000 0000 0000 0400 0000 1900 0000**

**fcff ffff ffff ffff a002 0000 0000 0000**

**0200 0000 0500 0000 7700 0000 0000 0000**

**a802 0000 0000 0000 0400 0000 1100 0000**

**fcff ffff ffff ffff b802 0000 0000 0000**

**0400 0000 1900 0000 fcff ffff ffff ffff**

**bf02 0000 0000 0000 0200 0000 0500 0000**

**7300 0000 0000 0000 c702 0000 0000 0000**

**0400 0000 1100 0000 fcff ffff ffff ffff**

**d702 0000 0000 0000 0400 0000 1900 0000**

**fcff ffff ffff ffff e302 0000 0000 0000**

**0200 0000 0e00 0000 fcff ffff ffff ffff**

**0203 0000 0000 0000 0200 0000 1400 0000**

**fcff ffff ffff ffff 0703 0000 0000 0000**

**0400 0000 1500 0000 fcff ffff ffff ffff**

**1903 0000 0000 0000 0400 0000 1500 0000**

**fcff ffff ffff ffff 3103 0000 0000 0000**

**0200 0000 0f00 0000 fcff ffff ffff ffff**

**3603 0000 0000 0000 0400 0000 1900 0000**

**fcff ffff ffff ffff 3d03 0000 0000 0000**

**0200 0000 0500 0000 7b00 0000 0000 0000**

**4503 0000 0000 0000 0400 0000 1100 0000**

**fcff ffff ffff ffff 5503 0000 0000 0000**

**0400 0000 1900 0000 fcff ffff ffff ffff**

**5c03 0000 0000 0000 0200 0000 0500 0000**

**7300 0000 0000 0000 6403 0000 0000 0000**

**0400 0000 1100 0000 fcff ffff ffff ffff**

**7403 0000 0000 0000 0400 0000 1900 0000**

**fcff ffff ffff ffff 8003 0000 0000 0000**

**0200 0000 0e00 0000 fcff ffff ffff ffff**

**ac03 0000 0000 0000 0200 0000 0e00 0000**

**fcff ffff ffff ffff d203 0000 0000 0000**

**0400 0000 1a00 0000 fcff ffff ffff ffff**

**dd03 0000 0000 0000 0200 0000 0e00 0000**

**fcff ffff ffff ffff 0804 0000 0000 0000**

**0200 0000 0400 0000 fcff ffff ffff ffff**

**0d04 0000 0000 0000 0400 0000 1c00 0000**

**fcff ffff ffff ffff 1404 0000 0000 0000**

**0200 0000 1d00 0000 fcff ffff ffff ffff**

**1b04 0000 0000 0000 0200 0000 0400 0000**

**fcff ffff ffff ffff 2204 0000 0000 0000**

**2a00 0000 1e00 0000 fcff ffff ffff ffff**

**2a04 0000 0000 0000 0400 0000 1f00 0000**

**fcff ffff ffff ffff 0000 0000 0000 0000**

**0100 0000 0200 0000 3104 0000 0000 0000**

**2000 0000 0000 0000 0200 0000 0200 0000**

**0000 0000 0000 0000 4000 0000 0000 0000**

**0200 0000 0200 0000 d803 0000 0000 0000**

**6000 0000 0000 0000 0200 0000 0200 0000**

**e803 0000 0000 0000 8000 0000 0000 0000**

**0200 0000 0200 0000 3104 0000 0000 0000**

**002e 7379 6d74 6162 002e 7374 7274 6162**

**002e 7368 7374 7274 6162 002e 7265 6c61**

**2e74 6578 7400 2e64 6174 6100 2e62 7373**

**002e 726f 6461 7461 002e 7265 6c61 2e69**

**6e69 745f 6172 7261 7900 2e63 6f6d 6d65**

**6e74 002e 6e6f 7465 2e47 4e55 2d73 7461**

**636b 002e 7265 6c61 2e65 685f 6672 616d**

**6500 0000 0000 0000 0000 0000 0000 0000**

**0000 0000 0000 0000 0000 0000 0000 0000**

**0000 0000 0000 0000 0000 0000 0000 0000**

**0000 0000 0000 0000 0000 0000 0000 0000**

**0000 0000 0000 0000 2000 0000 0100 0000**

**0600 0000 0000 0000 0000 0000 0000 0000**

**4000 0000 0000 0000 4604 0000 0000 0000**

**0000 0000 0000 0000 0100 0000 0000 0000**

**0000 0000 0000 0000 1b00 0000 0400 0000**

**4000 0000 0000 0000 0000 0000 0000 0000**

**100b 0000 0000 0000 b808 0000 0000 0000**

**0c00 0000 0100 0000 0800 0000 0000 0000**

**1800 0000 0000 0000 2600 0000 0100 0000**

**0300 0000 0000 0000 0000 0000 0000 0000**

**8604 0000 0000 0000 0000 0000 0000 0000**

**0000 0000 0000 0000 0100 0000 0000 0000**

**0000 0000 0000 0000 2c00 0000 0800 0000**

**0300 0000 0000 0000 0000 0000 0000 0000**

**8604 0000 0000 0000 0100 0000 0000 0000**

**0000 0000 0000 0000 0100 0000 0000 0000**

**0000 0000 0000 0000 3100 0000 0100 0000**

**0200 0000 0000 0000 0000 0000 0000 0000**

**8604 0000 0000 0000 8300 0000 0000 0000**

**0000 0000 0000 0000 0100 0000 0000 0000**

**0000 0000 0000 0000 3e00 0000 0e00 0000**

**0300 0000 0000 0000 0000 0000 0000 0000**

**1005 0000 0000 0000 0800 0000 0000 0000**

**0000 0000 0000 0000 0800 0000 0000 0000**

**0800 0000 0000 0000 3900 0000 0400 0000**

**4000 0000 0000 0000 0000 0000 0000 0000**

**c813 0000 0000 0000 1800 0000 0000 0000**

**0c00 0000 0600 0000 0800 0000 0000 0000**

**1800 0000 0000 0000 4a00 0000 0100 0000**

**3000 0000 0000 0000 0000 0000 0000 0000**

**1805 0000 0000 0000 2a00 0000 0000 0000**

**0000 0000 0000 0000 0100 0000 0000 0000**

**0100 0000 0000 0000 5300 0000 0100 0000**

**0000 0000 0000 0000 0000 0000 0000 0000**

**4205 0000 0000 0000 0000 0000 0000 0000**

**0000 0000 0000 0000 0100 0000 0000 0000**

**0000 0000 0000 0000 6800 0000 0100 0000**

**0200 0000 0000 0000 0000 0000 0000 0000**

**4805 0000 0000 0000 9800 0000 0000 0000**

**0000 0000 0000 0000 0800 0000 0000 0000**

**0000 0000 0000 0000 6300 0000 0400 0000**

**4000 0000 0000 0000 0000 0000 0000 0000**

**e013 0000 0000 0000 6000 0000 0000 0000**

**0c00 0000 0a00 0000 0800 0000 0000 0000**

**1800 0000 0000 0000 0100 0000 0200 0000**

**0000 0000 0000 0000 0000 0000 0000 0000**

**e005 0000 0000 0000 0003 0000 0000 0000**

**0d00 0000 0e00 0000 0800 0000 0000 0000**

**1800 0000 0000 0000 0900 0000 0300 0000**

**0000 0000 0000 0000 0000 0000 0000 0000**

**e008 0000 0000 0000 2f02 0000 0000 0000**

**0000 0000 0000 0000 0100 0000 0000 0000**

**0000 0000 0000 0000 1100 0000 0300 0000**

**0000 0000 0000 0000 0000 0000 0000 0000**

**4014 0000 0000 0000 7200 0000 0000 0000**

**0000 0000 0000 0000 0100 0000 0000 0000**

**0000 0000 0000 0000**

**Step 4:** Convert the assembly code to C++ code.

| **Blocks of Assembly Code** | **C++ Code** |
| --- | --- |
| main:  push %rbp  mov %rsp,%rbp  callq 3e1 <main+0x9>  mov $0x0,%eax  pop %rbp  retq | int main()  {  displayMenu();  } |
| push %rbp  mov %rsp,%rbp  sub $0x20,%rsp  mov %fs:0x28,%rax  mov %rax,-0x8(%rbp)  xor %eax,%eax  movl $0x1,-0x10(%rbp)  lea 0x0(%rip),%rsi # 25 <\_Z11displayMenuv+0x25>  lea 0x0(%rip),%rdi # 2c <\_Z11displayMenuv+0x2c>  callq 31 <\_Z11displayMenuv+0x31>  mov %rax,%rdx  mov 0x0(%rip),%rax # 3b <\_Z11displayMenuv+0x3b>  mov %rax,%rsi  mov %rdx,%rdi  callq 46 <\_Z11displayMenuv+0x46>  lea 0x0(%rip),%rsi # 4d <\_Z11displayMenuv+0x4d>  lea 0x0(%rip),%rdi # 54 <\_Z11displayMenuv+0x54>  callq 59 <\_Z11displayMenuv+0x59>  mov %rax,%rdx  mov 0x0(%rip),%rax # 63 <\_Z11displayMenuv+0x63>  mov %rax,%rsi  mov %rdx,%rdi  callq 6e <\_Z11displayMenuv+0x6e>  lea 0x0(%rip),%rsi # 75 <\_Z11displayMenuv+0x75>  lea 0x0(%rip),%rdi # 7c <\_Z11displayMenuv+0x7c>  callq 81 <\_Z11displayMenuv+0x81>  mov %rax,%rdx  mov 0x0(%rip),%rax # 8b <\_Z11displayMenuv+0x8b>  mov %rax,%rsi  mov %rdx,%rdi  callq 96 <\_Z11displayMenuv+0x96>  lea 0x0(%rip),%rsi # 9d <\_Z11displayMenuv+0x9d>  lea 0x0(%rip),%rdi # a4 <\_Z11displayMenuv+0xa4>  callq a9 <\_Z11displayMenuv+0xa9>  mov %rax,%rdx  mov 0x0(%rip),%rax # b3 <\_Z11displayMenuv+0xb3>  mov %rax,%rsi  mov %rdx,%rdi | void displayMenu() {  int x, y, selection, solution;  int displayLoop = 1;  cout << "----------------" << endl;  cout << "- 1)Add -" << endl;  cout << "- 2)Subtract -" << endl;  cout << "- 3)Multiply -" << endl;  cout << "- 4)Exit -" << endl;  cout << "----------------" << endl;  cin >> selection;  ....... |
| callq be <\_Z11displayMenuv+0xbe>  lea 0x0(%rip),%rsi # c5 <\_Z11displayMenuv+0xc5>  lea 0x0(%rip),%rdi # cc <\_Z11displayMenuv+0xcc>  callq d1 <\_Z11displayMenuv+0xd1>  mov %rax,%rdx  mov 0x0(%rip),%rax # db <\_Z11displayMenuv+0xdb>  mov %rax,%rsi  mov %rdx,%rdi  callq e6 <\_Z11displayMenuv+0xe6>  lea 0x0(%rip),%rsi # ed <\_Z11displayMenuv+0xed>  lea 0x0(%rip),%rdi # f4 <\_Z11displayMenuv+0xf4>  callq f9 <\_Z11displayMenuv+0xf9>  mov %rax,%rdx  mov 0x0(%rip),%rax # 103 <\_Z11displayMenuv+0x103>  mov %rax,%rsi  mov %rdx,%rdi  callq 10e <\_Z11displayMenuv+0x10e>  lea -0x14(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi # 11c <\_Z11displayMenuv+0x11c>  callq 121 <\_Z11displayMenuv+0x121>  lea 0x0(%rip),%rdi # 128 <\_Z11displayMenuv+0x128>  callq 12d <\_Z11displayMenuv+0x12d> | Continuation of previous block (couts and cin statement) |
| test %al,%al  je 193 <\_Z11displayMenuv+0x193>  mov $0x0,%esi  lea 0x0(%rip),%rdi # 13d <\_Z11displayMenuv+0x13d>  callq 142 <\_Z11displayMenuv+0x142>  mov $0xa,%edx  mov $0x7fffffff,%esi  lea 0x0(%rip),%rdi # 153 <\_Z11displayMenuv+0x153>  callq 158 <\_Z11displayMenuv+0x158>  lea 0x0(%rip),%rsi # 15f <\_Z11displayMenuv+0x15f>  lea 0x0(%rip),%rdi # 166 <\_Z11displayMenuv+0x166>  callq 16b <\_Z11displayMenuv+0x16b>  lea 0x0(%rip),%rsi # 172 <\_Z11displayMenuv+0x172>  lea 0x0(%rip),%rdi # 179 <\_Z11displayMenuv+0x179>  callq 17e <\_Z11displayMenuv+0x17e>  lea -0x14(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi # 18c <\_Z11displayMenuv+0x18c>  callq 191 <\_Z11displayMenuv+0x191> | //Security vulnerability existed on cin for selection variable, this block ensures in case of failure the buffer is restored  //Due to the new structure of displayMenu(), this also handles when users try to input out of range values for x or y and  //informs the user and goes back to the menu.  while (cin.fail()) {  cin.clear(); // clear input buffer to restore cin to a usable state  cin.ignore(INT\_MAX, '\n'); // ignore last input  cout << "You can only enter valid numbers.\n";  cout << "Enter a number.\n";  cin >> selection;  } |
| jmp 121 <\_Z11displayMenuv+0x121>  cmpl $0x0,-0x10(%rbp)  je 3c1 <\_Z11displayMenuv+0x3c1>  mov -0x14(%rbp),%eax  cmp $0x1,%eax  jne 24e <\_Z11displayMenuv+0x24e>  lea -0x1c(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi # 1b7 <\_Z11displayMenuv+0x1b7>  callq 1bc <\_Z11displayMenuv+0x1bc>  mov %rax,%rdx  lea -0x18(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 1ce <\_Z11displayMenuv+0x1ce>  mov -0x1c(%rbp),%edx  mov -0x18(%rbp),%eax  add %edx,%eax  mov %eax,-0xc(%rbp)  mov -0x1c(%rbp),%eax  mov %eax,%esi  lea 0x0(%rip),%rdi # 1e5 <\_Z11displayMenuv+0x1e5>  callq 1ea <\_Z11displayMenuv+0x1ea>  lea 0x0(%rip),%rsi # 1f1 <\_Z11displayMenuv+0x1f1>  mov %rax,%rdi  callq 1f9 <\_Z11displayMenuv+0x1f9>  mov %rax,%rdx  mov -0x18(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 209 <\_Z11displayMenuv+0x209>  lea 0x0(%rip),%rsi # 210 <\_Z11displayMenuv+0x210>  mov %rax,%rdi  callq 218 <\_Z11displayMenuv+0x218>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 228 <\_Z11displayMenuv+0x228>  mov %rax,%rdx  mov 0x0(%rip),%rax # 232 <\_Z11displayMenuv+0x232>  mov %rax,%rsi  mov %rdx,%rdi  callq 23d <\_Z11displayMenuv+0x23d>  movl $0x0,-0x10(%rbp)  callq 249 <\_Z11displayMenuv+0x249> | while (displayLoop) {  if (selection == 1) {  cin >> x >> y;  solution = x + y;  cout << x << " + " << y << " = " << solution << endl;  displayLoop = 0;  displayMenu();  } |
| jmpq 193 <\_Z11displayMenuv+0x193>  mov -0x14(%rbp),%eax  cmp $0x2,%eax  jne 2ec <\_Z11displayMenuv+0x2ec>  lea -0x1c(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi # 268 <\_Z11displayMenuv+0x268>  callq 26d <\_Z11displayMenuv+0x26d>  mov %rax,%rdx  lea -0x18(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 27f <\_Z11displayMenuv+0x27f>  mov -0x1c(%rbp),%edx  mov -0x18(%rbp),%eax  sub %eax,%edx  mov %edx,%eax  mov %eax,-0xc(%rbp)  mov -0x1c(%rbp),%eax  mov %eax,%esi  lea 0x0(%rip),%rdi # 298 <\_Z11displayMenuv+0x298>  callq 29d <\_Z11displayMenuv+0x29d>  lea 0x0(%rip),%rsi # 2a4 <\_Z11displayMenuv+0x2a4>  mov %rax,%rdi  callq 2ac <\_Z11displayMenuv+0x2ac>  mov %rax,%rdx  mov -0x18(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 2bc <\_Z11displayMenuv+0x2bc>  lea 0x0(%rip),%rsi # 2c3 <\_Z11displayMenuv+0x2c3>  mov %rax,%rdi  callq 2cb <\_Z11displayMenuv+0x2cb>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 2db <\_Z11displayMenuv+0x2db>  movl $0x0,-0x10(%rbp)  callq 2e7 <\_Z11displayMenuv+0x2e7> | else if (selection == 2) {  cin >> x >> y;  solution = x - y;  cout << x << " - " << y << " = " << solution;  displayLoop = 0;  displayMenu();  } |
| jmpq 193 <\_Z11displayMenuv+0x193>  mov -0x14(%rbp),%eax  cmp $0x3,%eax  jne 389 <\_Z11displayMenuv+0x389>  lea -0x1c(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi # 306 <\_Z11displayMenuv+0x306>  callq 30b <\_Z11displayMenuv+0x30b>  mov %rax,%rdx  lea -0x18(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 31d <\_Z11displayMenuv+0x31d>  mov -0x1c(%rbp),%edx  mov -0x18(%rbp),%eax  imul %edx,%eax  mov %eax,-0xc(%rbp)  mov -0x1c(%rbp),%eax  mov %eax,%esi  lea 0x0(%rip),%rdi # 335 <\_Z11displayMenuv+0x335>  callq 33a <\_Z11displayMenuv+0x33a>  lea 0x0(%rip),%rsi # 341 <\_Z11displayMenuv+0x341>  mov %rax,%rdi  callq 349 <\_Z11displayMenuv+0x349>  mov %rax,%rdx  mov -0x18(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 359 <\_Z11displayMenuv+0x359>  lea 0x0(%rip),%rsi # 360 <\_Z11displayMenuv+0x360>  mov %rax,%rdi  callq 368 <\_Z11displayMenuv+0x368>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 378 <\_Z11displayMenuv+0x378>  movl $0x0,-0x10(%rbp)  callq 384 <\_Z11displayMenuv+0x384> | else if (selection == 3) {  cin >> x >> y;  solution = x \* y;  cout << x << " \* " << y << " = " << solution;  displayLoop = 0;  displayMenu();  } |
| jmpq 193 <\_Z11displayMenuv+0x193>  mov -0x14(%rbp),%eax  cmp $0x3,%eax  jle 39d <\_Z11displayMenuv+0x39d>  movl $0x0,-0x10(%rbp)  jmpq 193 <\_Z11displayMenuv+0x193>  mov -0x14(%rbp),%eax  test %eax,%eax  jg 3b5 <\_Z11displayMenuv+0x3b5>  movl $0x0,-0x10(%rbp)  callq 3b0 <\_Z11displayMenuv+0x3b0>  jmpq 193 <\_Z11displayMenuv+0x193>  movl $0x1,-0x10(%rbp)  jmpq 193 <\_Z11displayMenuv+0x193>  nop  mov -0x8(%rbp),%rax  xor %fs:0x28,%rax  je 3d6 <\_Z11displayMenuv+0x3d6>  callq 3d6 <\_Z11displayMenuv+0x3d6>  leaveq  retq | //Below are extra measures to validate 'selection' input that may cause issues  else if (selection >= 4){  displayLoop = 0;  }  else if (selection < 1) {  displayLoop = 0;  displayMenu();  }  else  displayLoop = 1;  }  } |