## Middle Square Weyl Sequence PRNG

**Example**

Suppose "x = 0xace983fe671dbd09". Then "x" is a 64-bit number with the following bits:

1010110011101001100000111111111001100111000111011011110100001001

When that number is squared, it becomes the 128-bit number 0x74ca9e5f63b6047f6a65456d9da04a51, or in binary:

01110100110010101001111001011111011000111011011000000100011111110110101001100101010001010110110110011101101000000100101001010001

But remember, "x" is a 64-bit number, so in our C code, only the bottom 64-bits are returned from that 128-bit number. So "x" is really 0x6a65456d9da04a51, or in binary:

0110101001100101010001010110110110011101101000000100101001010001

But the bits "01101010011001010100010101101101" are the 3rd 32-bits of the 128-bit number that was the result of squaring "x" (see above). They are the "middle" 32-bits that we're after. So, we're going to do something rather clever. We're going to swap the upper 32-bits with the lower, then return the lower 32-bits.

Effectively, what we're doing is "ABCD" -> "CDAB", then returning "AB". We do this via bit-shifting. So, starting with:

0110101001100101010001010110110110011101101000000100101001010001

First, we bitshift the 64-bit number right 32-bits:

0110101001100101010001010110110110011101101000000100101001010001 >> 32

= 0000000000000000000000000000000001101010011001010100010101101101

Then we bitshift "x" left 32-bits:

0110101001100101010001010110110110011101101000000100101001010001 << 32

= 1001110110100000010010100101000100000000000000000000000000000000

Now we logically "or" them together:

0000000000000000000000000000000001101010011001010100010101101101 |

1001110110100000010010100101000100000000000000000000000000000000

|================================================================

1001110110100000010010100101000101101010011001010100010101101101

See the swap? Now, due to the function return width, we return the lower 32-bits as our random number, which is 01101010011001010100010101101101, or 1785021805 in decimal. We've arrived at our goal.