Phase 1: The answer is a string hidden in the binary file. After using the "strings" command we can clearly see the string "`Why make trillions when we could make... billions?`"

Phase 2: First we infer that 6 integers are required as the input format. Then, the first two values of these integers can be inferred in the assembly code. Then, a loop occurs which corresponds to a[i] = a[i=1] + a[i-2]. It's the Fibonacci sequence.

Phase 3: Just like in the second phase, there is a required input format which in my case was %d %c %d. After determining the input format, I randomly gave 3 values and stepped over each assembly instruction where my values were compared with the desired output. After printing the values of the desired input, I transformed these hexadecimal representations with the corresponding values in integers and characters.

Phase 4: We determine the input format. Then, we see that func4 is a recursive function and after computing the values, we see that the inputs are (4,80)

Phase 5: After determining the input format (which are 6 characters) We see that each character is AND'ed with 0xf which corresponds to the last 4 bits. The values of each characters last 4 bits are added and compared to the value of 23, if equal, successfully exits the phase. This means there is more than one solution. I picked aaappc.

Phase 6: After determining the input format, we see that the values must be 1, 2, 3, 4 ,5, 6. Then we see that 6 nodes are created with these inputs. After finding the memory addresses of these nodes (It can be obtained by dereferencing the "next".) I noted each node's memory address. Then we see that there is a loop which compares each nodes value (not to be confused with the integer inside it). After figuring out the comparison method, we see that the values are 6, 2, 5, 4, 3, 1.