Report Question 1:

runtime for cargo run:

k: 0

F_k at 0: 0

Time it took: 911.3µs

k: 1

F_k at 1: 1

Time it took: 849.3µs

k: 2

F_k at 2: 1

Time it took: 589.9µs

k: 3

F_k at 3: 2

Time it took: 544.5µs

k: 4

F_k at 4: 3

Time it took: 632.3µs

k: 5

F_k at 5: 5

Time it took: 593.2µs

k: 6

F_k at 6: 8

Time it took: 595.3µs

k: 7

F_k at 7: 13

Time it took: 590.1µs

k: 20

F_k at 20: 6765

Time it took: 726.4µs

k: 21

F_k at 21: 10946

Time it took: 1.6763ms

k: 22

F_k at 22: 17711

Time it took: 1.7527ms

k: 23

F_k at 23: 28657

Time it took: 1.8443ms

k: 24

F_k at 24: 46368

Time it took: 1.2847ms

k: 44

F_k at 44: 701408733

Time it took: 8.5684472s

k: 45

F_k at 45: 1134903170

Time it took: 13.686929s

k: 46

F_k at 46: 1836311903

Time it took: 22.3547562s

k: 47

F_k at 47: 2971215073

Time it took: 35.7867542s

k: 48

F_k at 48: 4807526976

Time it took: 61.2331569s

runtime for cargo run --release:

k: 0

F_k at 0: 0

Time it took: 1.2701ms

k: 1

F_k at 1: 1

Time it took: 1.1278ms

k: 2

F_k at 2: 1

Time it took: 528.6µs

k: 3

F_k at 3: 2

Time it took: 697.6µs

k: 4

F_k at 4: 3

Time it took: 596µs

k: 5

F_k at 5: 5

Time it took: 682.2µs

k: 6

F_k at 6: 8

Time it took: 829.2µs

k: 7

F_k at 7: 13

Time it took: 964.3µs

k: 20

F_k at 20: 6765

Time it took: 1.7991ms

k: 21

F_k at 21: 10946

Time it took: 3.1511ms

k: 22

F_k at 22: 17711

Time it took: 1.7803ms

k: 23

F_k at 23: 28657

Time it took: 1.9184ms

k: 24

F_k at 24: 46368

Time it took: 1.0013ms

k: 44

F_k at 44: 701408733

Time it took: 4.0898488s

k: 45

F_k at 45: 1134903170

Time it took: 7.116609s

k: 46

F_k at 46: 1836311903

Time it took: 10.9948321s

k: 47

F_k at 47: 2971215073

Time it took: 17.8653341s

k: 48

F_k at 48: 4807526976

Time it took: 29.6260071s

Running through cargo run --release gives significantly quicker run times when running high values of k. At low values of k, however, cargo run gives quicker runtime. This shift can be seen around k=21, where before k=21 using cargo run is faster, but above k=21 using cargo run --release is faster. This is most evident at high values of k, for example at k=48 cargo run --release ran in 29.6s, but cargo run took 61.2s

to run at k=48. The release version took roughly half as long at high values of k, but at lower values the non release version can take half as long as the release version.

Report Question 2:

running with cargo run using type u8 throws an error. Specifically, the main function panicked since it attempted to add with overflow, this happened since u8 cannot store the large numbers that this function requires, so to complete the task it would cause integer overflow, which the compiler noticed and caught. When running with cargo run --release, the compiler does not check for integer overflow issues. Therefore, the code runs as normal, but the output is not correct. Numbers that are output are only as high as u8 allows for, so 255 maximum, which means that when the addition produces a number over 255, it wraps back around and starts from 0 again. When using u128, the memory for the value is higher which allows for correct addition of higher numbers.

Report Question 3:

here, integer overflow was not an issue since my sum variable uses type u128. This can store the maximum possible value that Rust can represent, so it should be able to store any value produced from the sum of squares from 1 to the largest possible u32 value.