Homework Number: 7

Name: Andrew Wu

ECN Login: wu1795

Due Date: 3/5/24

Input:

Boku no Kokoro no Yabai Yatsu is the greatest romance, slice of life manga I've ever read. It is a series of constant progress that respects the reader's time and trusts them to read between the lines - Characters make mistakes and learn from them. Misunderstandings are never used to pad out the story, and never feel cheap. Progress is never undone. It is one of the most fully realized depictions of the liminal space between two young people as they begin to fall in love - The roller coaster between bubbly feelings and crippling cringe that is first love is so difficult to portray. I've never encountered another manga that has managed to capture this specific feeling so accurately and with so much detail.

Output:

84f353348a552229554fba7ba822005edcb6bca2fac8cf1735d53ae9e2915aa2e625f6d3cfa0106c8707ff0004d3ce95281b47b851b380ef91c86d2fb0e58b28

At the top of my code, I initialize the 8 words used in the initialization vector, the 80 K constants, and a bitvector variable that stores the 80 K constants in my init function to use in various areas later in my program. In my hashing function, I start out by opening and reading the input file and turn the input into a bitvector, while also opening and writing the output file. I then follow the steps of the hashing function. I first pad the message to make sure it is a multiple of 1024 bits, while also accounting for the last 128 bits being used to store the length of the message. To do this, I append a 1 to the end of the input message and then pad zeros until the new length is correct, and then note down the number of zeros used. I then combine the new padded message with the original message to push a combined message into the hashing algorithm. I then create the message schedule for the 1024 bit message. I first generate the first 16 words of the 80 word message schedule, and then I calculate the rest of the words using the equations provided on page 43 and 44 of the lecture notes. I then proceed to encode the round function to hash the message using the message schedule. I initialize each of the 8 initial words of the initialization vector into letter variables to use them in calculations. Then, for 80 rounds, I create and use the variables found on page 45 and 46. These variables and bitvectors are then used to calculate the output for each round and add them to the corresponding has buffers at the beginning of the round-based processing. This process repeats until the entire message has been hashed. Once the message is completely hashed, I combine the resulting 8 64 bit words together and write the combination into the output file in hex. A lot of my code has been adapted from the lecture code of sha256.py. As stated in the beginning, I have moved the k constants and the 8 words in the initialization vector used for sha512 to the init function, allowing me to better store and utilize the variables when needed.