Riya Danait

CSCE 1030 Computer Science I – Section 002

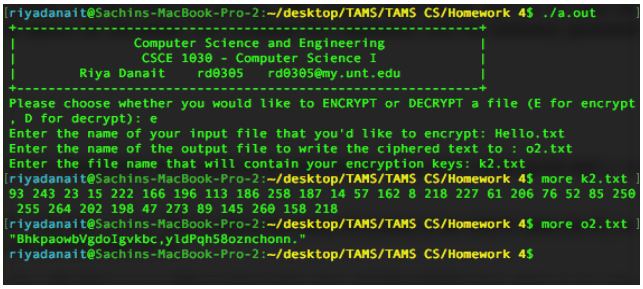
Professor Helsing

November 3rd, 2017

Algorithm for Homework 4

Calculations:

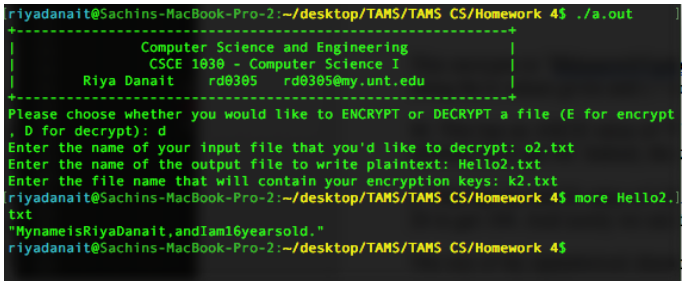
Let's try this encryption-decryption system with the following message: "My name is Riya Danait, and I am 16 years old." We will note that punctuation is kept the same and spaces are discarded. Let's evaluate this expression with the randomly generated k values from the program:



This encrypts to "BhkpaowbVgdoIgvkbc,yldPqh58oznchonn." If we perform the calculations using the k values given and c = m + (k % 26), we get:

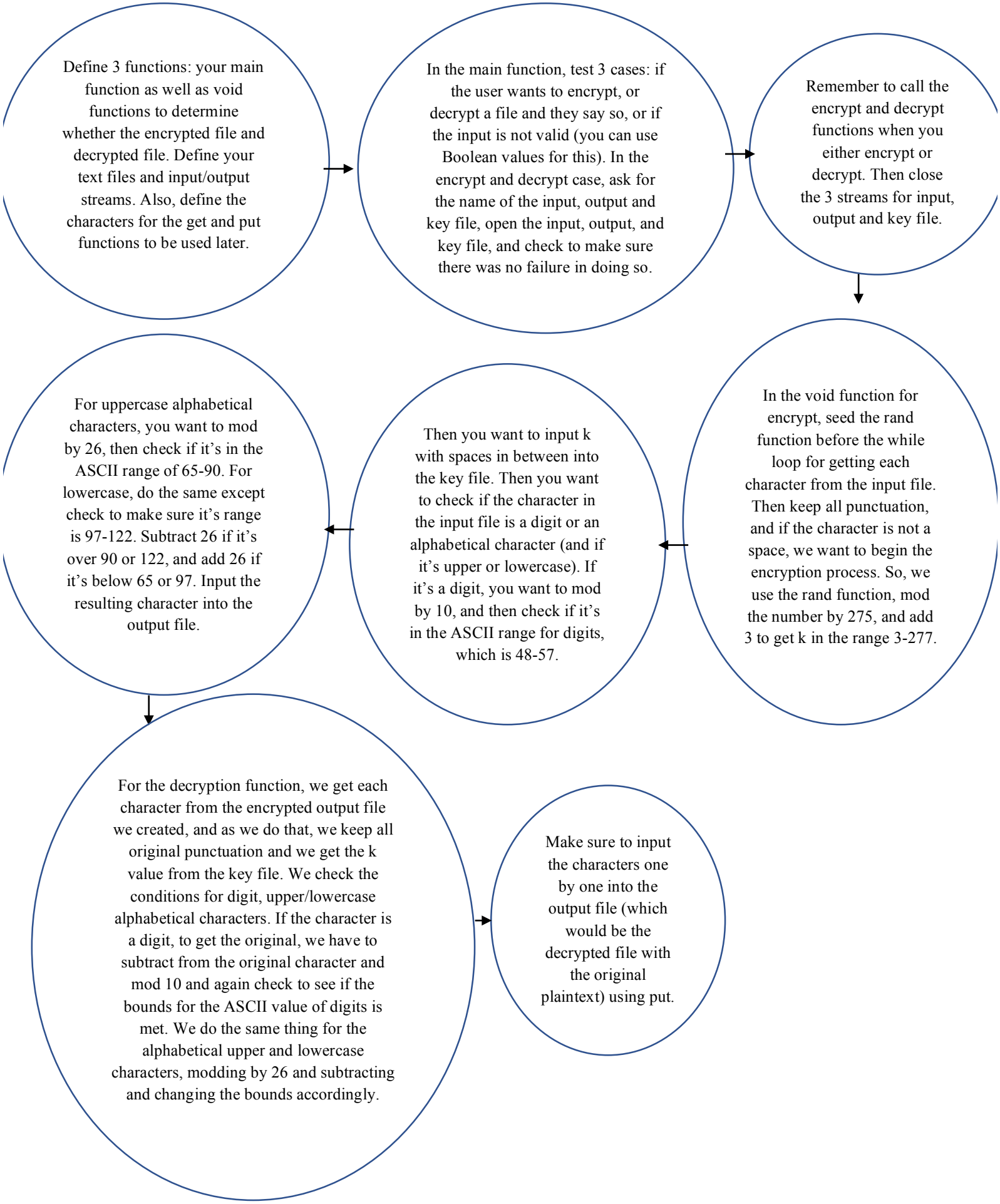
M: This has an ASCII value of 77. 77 + (93 % 26) = c. c = 92. This is above the value 90 (which is the ASCII value for Z), so we subtract 26 to get 66. Indeed, the character B has an ASCII value of 66. **y**: This has an ASCII value of 121. 121 + (243 % 26) = c. c = 130. This is above the value of 122 (ASCII value of z), so we subtract 26 to get 104. And surely we see that the character h has an ASCII value of 104. The rest of the alphabetical characters can be figured out in the same way. For digits, we can use c = m + (k % 10). So, in this example, we had 16: **1**: This has an ASCII value of 49. 49 + (264 % 10) = c. c = 53. This is below the ASCII value of 57. And indeed, we see 5 has an ASCII value of 53. **6**: This has an ASCII value of 54. 54 + (202 % 10) = c. c = 56. We see that 8 has an ASCII value of 56.

So, the encryption algorithm works. Let's try decryption. We can perform the decryption by using m = c – (k % 26) for alphabetical characters, and m = c – (k % 10) for digits and adjust/wrap around where needed.



Let's choose a different character, say k that was encrypted from n in name, to decrypt:

k: This has an ASCII value of 107. m = 107 – (23 % 26). m = 84, which is lower than 97, so we add 26 to get 110. This is indeed to ASCII value for n. **V**: This has an ASCII value of 86. m = 86 – (186 % 26). m = 82. This is less than 90, so we see that an ASCII value of 82 yields R. The rest of the alphabetical characters can be figured out in the same way. Let's look at **5**: This has an ASCII value of 53. m = 53 – (264 % 10). m = 49. 1 has an ASCII value of 49. Then **8**: This has an ASCII value of 56. m = 56 – (202 % 10). m = 54. And 6 has an ASCII value of 54. So the encryption-decryption system works for alphanumeric characters, keeps punctuation, and discards the spaces in both instances.


Define 3 functions: your main function as well as void functions to determine whether the encrypted file and decrypted file. Define your text files and input/output streams. Also, define the characters for the get and put functions to be used later.
In the main function, test 3 cases: if the user wants to encrypt, or decrypt a file and they say so, or if the input is not valid (you can use Boolean values for this). In the encrypt and decrypt case, ask for the name of the input, output and key file, open the input, output, and key file, and check to make sure there was no failure in doing so.
Remember to call the encrypt and decrypt functions when you either encrypt or decrypt. Then close the 3 streams for input, output and key file.
In the void function for encrypt, seed the rand function before the while loop for getting each character from the input file. Then keep all punctuation, and if the character is not a space, we want to begin the encryption process. So, we use the rand function, mod the number by 275, and add 3 to get k in the range 3-277.
Then you want to input k with spaces in between into the key file. Then you want to check if the character in the input file is a digit or an alphabetical character (and if it's upper or lowercase). If it's a digit, you want to mod by 10, and then check if it's in the ASCII range for digits, which is 48-57.
For uppercase alphabetical characters, you want to mod by 26, then check if it's in the ASCII range of 65-90. For lowercase, do the same except check to make sure it's range is 97-122. Subtract 26 if it's over 90 or 122, and add 26 if it's below 65 or 97. Input the resulting character into the output file.
For the decryption function, we get each character from the encrypted output file we created, and as we do that, we keep all original punctuation and we get the k value from the key file. We check the conditions for digit, upper/lowercase alphabetical characters. If the character is a digit, to get the original, we have to subtract from the original character and mod 10 and again check to see if the bounds for the ASCII value of digits is met. We do the same thing for the alphabetical upper and lowercase characters, modding by 26 and subtracting and changing the bounds accordingly.
Make sure to input the characters one by one into the output file (which would be the decrypted file with the original plaintext) using put.