|  |  |
| --- | --- |
| Cipher | Error handling |
| Caesar  Requirement:  Plaintext->all small letter alphabet  Key->must be integer | * If input plaintext contains non-alphabet, will just output spacebar (‘ ’) for that letter without doing any operations {fixed: input plaintext only contains alphabet} * If input plaintext contains big letter, it will be treated as small letter (MEETMEAFTER = meetmeafter) {fixed: input plaintext only considered small letter alphabet} * If input key wasn’t in integer form(14.0 will be ok, but 14.1 will be not ok), the invalid part will be dropped (1feafef4 = 14) {fixed: input key will only be integer form} |
| Playfair  Requirement:  Plaintext->all small letter alphabet  Key->all big letter alphabet | * If input plaintext contains non-alphabet, will just output spacebar (‘ ‘) for that letter without doing any operations {fixed: input plaintext only contains alphabet} * If input plaintext length is odd, it will append the plaintext with ‘X’ (PLAINTEXT = PLAINTEXTX) {fixed: length of input plaintext will only be even} * If input plaintext contains pair with same letter, it will add a “X” after the first letter (AABB (plaintext) => AXABXB (plaintext)) {fixed: input plaintext will not contain pair with same letter} * If input plaintext contains big alphabet, the letter will be treated as small letter (HIDEINTHESTUMP = hideinthestump) {fixed: input plaintext will only contain small alphabet} * If input key contains non-alphabet, the letter will just get dropped (PL1A3YF2AIR will be treated as PLAYFAIR) {fixed: input key will only contain alphabet} * If input key contains small alphabet, the letter will be treated as big letter (playfair = PLAYFAIR) {fixed: input key will only contain big letter alphabet} * The letter ‘J’ will be treat as ‘I’ in plaintext and key (JACK = IACK) |
| Vernam  Requirement:  Plaintext->all small letter alphabet  Key->all big letter alphabet | * If input plaintext contains non-alphabet, will just output spacebar (‘ ‘) for that letter without doing any operations {fixed: input plaintext only contains alphabet} * If input plaintext contains big alphabet, the letter will be treated as small letter (HELO = helo) {fixed: input plaintext only contains small alphabet} * If input key contains small alphabet, the letter will be treated as big letter (xmcl = XMCL) {fixed: input key only contains big alphabet} * If input key contains non-alphabet, the letter will just get dropped (PL1A3YF2AIR will be treated as PLAYFAIR) {fixed: input key only contains alphabet} * If input key contains no alphabet, the cipher text will be all ‘A’ (as plaintext Xor itself) {fixed: input key only contains alphabet} |
| Row transposition  Requirement:  Plaintext-> all small letter alphabet | non alphabet  Key->all digit | * If input plaintext contains big alphabet, the letter will be treated as small letter (ATTACKPOST123PONE = attackpost123pone) {fixed: input plaintext only contains small letter alphabet} * If input plaintext can’t get completely divide by the length of key, the rest will fill in with ‘X’, (key = 123, plain = ABCA => key = 123, plain = ABCAXX) {fixed: this is a condition that wasn’t a bug} * If input key contains non digit, it will get dropped (123456effeafg = 123456) {fixed: input key only contains digit} * If input key digit are missing, it will concatenate the digit (142578 = 142356)   {fixed: this situation will not happen } |
| Rail fence cipher  Requirement:  Plaintext->all small letter alphabet | non alphabet  Key->all digit | * If input plaintext contains big alphabet, the letter will be treated as small letter (ATTACKPOST123PONE = attackpost123pone) {fixed: input plaintext only contains small letter alphabet}   If input key value is larger than the input plaintext, it will just treated as the length of plaintext (plaintext = ‘HELO’; key = 12 => plaintext = ‘HELO’; key = 4) {fixed: this situation will not happen}   * If input key contains non digit, it will get dropped (2edff = 2) {fixed: input key will only contain digit} |

Other error:

Not enough argv => no action

Invalid cipher => no action