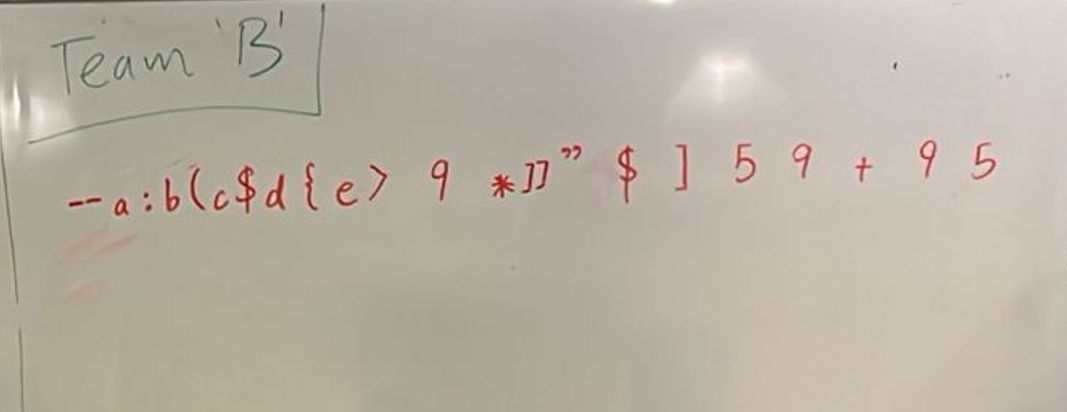
**TEAM B’s BOMBASTIC ENCRYPTION ALGORITHM 💪**



**Pre-encryption work:**

Create an “alphabet” (key-value pairs of each normal character in our language to another character). For the purpose of our algorithm and in order to make our cipher more difficult to crack, we created an alphabet that was case-sensitive i.e. mapped characters a to z and A to Z distinctively. In the interest of time, to create a distinct alphabet-to-new-character mapping of these 56 characters quickly, we used ChatGPT. One distinction we made in our new alphabet for our ciphered text was that it would not contain any characters in the normal English alphabet, and would only be composed of symbols and numbers.

**Encryption Algorithm:**

1. Given an input text, treat it as a String of n characters from the normal English alphabet (a to z and A to Z), with each character corresponding to either an even index or odd index on that String.
2. Considering the position of each character in the normal English language as an index, shift characters in the input string on even indices two positions backwards and replace it with that position’s character. For example, character ‘a’ on position 4 shifted two positions backwards means it is replaced with character ‘y’. Similarly, character ‘L’ on position 0 will be replaced by character 'J’. (Note: maintain cases (uppercase/lowercase) of each character).
3. For characters in the input string on odd indices, shift them two positions forward and replace it with that position’s character. For example, character ‘a’ on position 1 would be replaced with character ‘c’ and character ‘L’ on position 5 would be replaced with character 'N’.
4. For distinguishing separate words, replace every space character between words in the input string with this symbol ‘]]’.
5. Now using the new alphabet (new key-value pairs) that we created, replace these the characters with the corresponding symbols.
6. Finally, generate a random integer value (using any random number generating algorithm). Let this be X.
7. Now repeat this X times: generate another random integer value between 0 and n – 1 and insert a character not present in your new alphabet (i.e. any normal English language character will do since our new alphabet is composed only of numbers and symbols) at that position between the characters in the string obtained from Step 4. For example, if X was 5 and our original string was “12345678”, then after this step it may look like “1a2b3c4d5e678”.

The string obtained from Step 7 is our encrypted string.

**Demonstration:**

**Input string:** “Pakistan Zindabad”

**Steps 2 & 3:** “Pakistan Zindabad” 🡪 “Rymgurcl Bgpbczcb”

**Step 4:** “Pakistan Zindabad” 🡪 “Rymgurcl]]Bgpbczcb”

**Step 5:** “Rymgurcl]]Bgpbczcb” 🡪 “--:(${>9\*]]”$]59+95” (if our new alphabet looked like the one below)

a - ' b - 5 c - 9 d - # e - & f - % g - $ h - @ i - ! j - ?

k - ^ l - \* m - ( n - ) o - [ p - ] q - < r - > s - / t - \

u - { v - } w - | x - ; y - : z - + A - ` B - " C - [ D - ]

E - { F - } G - | H - ; I - : J - = K - , L - . M - < N - >

O - ? P - / Q - \ R - - S - = T - \_ U - + V - ( W - ) X - {

Y - } Z - [

**Step 6:** X = 5

**Step 7:** “--:(${>9\*]]”$]59+95” 🡪 “--a:b(c$d{e>9\*]]”$]59+95”

**Key observations:**

* Difficult to crack since standard key-value mapping cannot be deduced immediately i.e. “a” maps to “:” and “9” in the same string due to alternative shifting of odd and even index positions.
* Number of random characters added makes computations for cracking on hacker’s end even more tedious.
* Since the random characters added at Step 7 depend on the characters not used in the alphabet, if different systems would like to use this encryption algorithm with their own different randomly-generated new alphabets too, then that would add another layer of security and difficulty in cracking the cipher.
* Only if the key is intercepted, then the hacker can deduced the original data. Otherwise, our two-step encryption scheme protects us from that.
* Although random useless characters at Step 7 makes our cipher more difficult to crack, storing this and transferring these extra useless bits each time makes data sizes large for sending and storing.

**Team B:**

|  |  |
| --- | --- |
| Asad Virani | 22787 |
| Khizer Hayat Qureshi | 23000 |
| Sara Akbar Ebrahim | 22967 |
| Muhammad Ali Akbar | 22950 |
| Ashnah Khalid Khan | 22889 |
| Muhammad Rouhan Farooqui | 22819 |
| Syed zaid bin haris | 22868 |
| Aamna Salman | 22782 |
| Alliya Parvez | 22970 |
| Fatima Mustafa | 22996 |
| Zawiyar Khan | 22972 |
| Ahsan Ahmed | 22812 |
| Muhammad Zaiyan Soorty | 23055 |
| Ahsant Shahid | 22974 |
| Nabeel Mustafa | 22773 |
| Bilal Ahmad | 22867 |
| Amna Anjum | 22818 |
| Saad Riazuddin Mahmood | 22803 |
| Danish Badar Qureshi | 22890 |
| Shaz Shoaib | 21554 |
| Dania Ahmed | 22795 |
| Syeda Maham Jafri | 22796 |
| Fatimah Rashid | 22733 |
| Muhammad Rasib Nadeem | 22976 |
| Hasaan Ahmed Saeed | 22955 |
| Muhammad Arsalan Danyal | 23085 |
| Shehryar Hassan | 22750 |
| Ahmed Raza | 22785 |
| Warda Fatima Syed | 22832 |