

OUTPUTS FOR THE CODE SUBMITTED

1. Immediate output when the code is run:

```
Output
Choose Cipher Algorithm:
1. Caesar Cipher
2. Atbash Cipher
3. Augustus Cipher
4. Affine Cipher
5. Vigenère Cipher
6. Gronsfield Cipher
7. Beaufort Cipher
8. Autokey Cipher
9. Hill Cipher
10. Rail Fence Cipher
11. Ngram Cipher
12. Route Cipher
13. Myszkowski Cipher
Enter the number of the cipher you want to use: |
```

2. Caeser Cipher:

main.c

Share

Run

```
706     case 10:
707         printf("Enter the number of rails for Rail Fence Cipher
708             : ");
709         scanf("%d", &rails_count);
710         rail_fence_encode(message, rails_count, encoded_output
711             );
712         rail_fence_decode(encoded_output, rails_count,
713             decoded_output);
714         break;
715     case 11:
716         printf("Enter the value of n for Ngram Cipher: ");
717         scanf("%d", &chunk_size);
718         ngram_encode(message, chunk_size, encoded_output);
719         ngram_decode(encoded_output, chunk_size, decoded_output
720             );
721         break;
722     case 12:
723         printf("Enter the size of the matrix for Route Cipher:
724             ");
725         scanf("%d", &rails_count);
726         route_encode(message, rails_count, encoded_output);
727         route_decode(encoded_output, rails_count,
728             decoded_output);
729         break;
730     case 13:
731         printf("Enter the key for Myszkowski Cipher: ");
```

Output

Clear

```
Choose Cipher Algorithm:
1. Caesar Cipher
2. Atbash Cipher
3. Augustus Cipher
4. Affine Cipher
5. Vigenère Cipher
6. Gronsfield Cipher
7. Beaufort Cipher
8. Autokey Cipher
9. Hill Cipher
10. Rail Fence Cipher
11. Ngram Cipher
12. Route Cipher
13. Myszkowski Cipher
Enter the number of the cipher you want to use: 1
Enter the message: hello
Enter the rotation value for Caesar Cipher: 3
Encrypted: koor
Decrypted: hello

=== Code Execution Successful ===
```

Windows Update

Restart to install the latest Windows updates

New updates are ready for you. Restart now to get them on and plugged in, and we'll restart your PC during your active hours. Or, pick a time to restart.

Restart now

Pick a time

3. Atbash Cipher:

main.c	Output
<pre>706 case 10: 707 printf("Enter the number of rails for Rail Fence Cipher : "); 708 scanf("%d", &rails_count); 709 rail_fence_encode(message, rails_count, encoded_output); 710 rail_fence_decode(encoded_output, rails_count, decoded_output); 711 break; 712 case 11: 713 printf("Enter the value of n for Ngram Cipher: "); 714 scanf("%d", &chunk_size); 715 ngram_encode(message, chunk_size, encoded_output); 716 ngram_decode(encoded_output, chunk_size, decoded_output); 717 break; 718 case 12: 719 printf("Enter the size of the matrix for Route Cipher: "); 720 scanf("%d", &rails_count); 721 route_encode(message, rails_count, encoded_output); 722 route_decode(encoded_output, rails_count, decoded_output); 723 break; 724 case 13: 725 printf("Enter the key for Myszkowski Cipher: ");</pre>	<pre>Choose Cipher Algorithm: 1. Caesar Cipher 2. Atbash Cipher 3. Augustus Cipher 4. Affine Cipher 5. Vigenère Cipher 6. Gronsfeld Cipher 7. Beaufort Cipher 8. Autokey Cipher 9. Hill Cipher 10. Rail Fence Cipher 11. Ngram Cipher 12. Route Cipher 13. Myszkowski Cipher Enter the number of the cipher you want to use: 2 Enter the message: good morning Encrypted: tllw nlimrmt Decrypted: good morning === Code Execution Successful ===</pre>

4. Augustus Cipher:

main.c	Output
<pre>706 case 10: 707 printf("Enter the number of rails for Rail Fence Cipher : "); 708 scanf("%d", &rails_count); 709 rail_fence_encode(message, rails_count, encoded_output); 710 rail_fence_decode(encoded_output, rails_count, decoded_output); 711 break; 712 case 11: 713 printf("Enter the value of n for Ngram Cipher: "); 714 scanf("%d", &chunk_size); 715 ngram_encode(message, chunk_size, encoded_output); 716 ngram_decode(encoded_output, chunk_size, decoded_output); 717 break; 718 case 12: 719 printf("Enter the size of the matrix for Route Cipher: "); 720 scanf("%d", &rails_count); 721 route_encode(message, rails_count, encoded_output); 722 route_decode(encoded_output, rails_count, decoded_output); 723 break; 724 case 13: 725 printf("Enter the key for Myszkowski Cipher: ");</pre>	<pre>Choose Cipher Algorithm: 1. Caesar Cipher 2. Atbash Cipher 3. Augustus Cipher 4. Affine Cipher 5. Vigenère Cipher 6. Gronsfeld Cipher 7. Beaufort Cipher 8. Autokey Cipher 9. Hill Cipher 10. Rail Fence Cipher 11. Ngram Cipher 12. Route Cipher 13. Myszkowski Cipher Enter the number of the cipher you want to use: 3 Enter the message: shiv nadar university Encrypted: tijw obebs vojwfstjuz Decrypted: shiv nadar university === Code Execution Successful ===</pre>

5. Affine Cipher:

```
main.c  [Icons] [Run] [Share] [Clear]

437 }
438 clean_message[clean_pos] = '\0';
439
440 for (int pos = 0; pos < clean_pos; pos += chunk_size) {
441     int offset = 5; // Fixed shift value for easiness
442
443     for (int j = 0; j < chunk_size; j++) {
444         encoded_output[output_pos++] = ((clean_message[pos + j]
445             - 'A' + offset) % 26) + 'A';
446     }
447     if (pos + chunk_size < clean_pos) {
448         encoded_output[output_pos++] = ' ';
449     }
450 }
451 encoded_output[output_pos] = '\0';
452 }
453
454 void ngram_decode(char *encoded_message, int chunk_size, char
    *decoded_output) {
455     // Removing spaces
456     char clean_encoded[1000];
457     int clean_pos = 0;
458
459     for (int pos = 0; encoded_message[pos] != '\0'; pos++) {
```

Choose Cipher Algorithm:
1. Caesar Cipher
2. Atbash Cipher
3. Augustus Cipher
4. Affine Cipher
5. Vigenère Cipher
6. Gronsfeld Cipher
7. Beaufort Cipher
8. Autokey Cipher
9. Hill Cipher
10. Rail Fence Cipher
11. Ngram Cipher
12. Route Cipher
13. Myszkowski Cipher
Enter the number of the cipher you want to use: 4
Enter the message: afternoon
Enter the multiplier for Affine Cipher: 5
Enter the additive constant for Affine Cipher: 8
Encrypted: ihzcpvaav
Decrypted: afternoon

=== Code Execution Successful ===

6. Vignere Cipher:

```
main.c  [Icons] [Run] [Share] [Clear]

437 }
438 clean_message[clean_pos] = '\0';
439
440 for (int pos = 0; pos < clean_pos; pos += chunk_size) {
441     int offset = 5; // Fixed shift value for easiness
442
443     for (int j = 0; j < chunk_size; j++) {
444         encoded_output[output_pos++] = ((clean_message[pos + j]
445             - 'A' + offset) % 26) + 'A';
446     }
447     if (pos + chunk_size < clean_pos) {
448         encoded_output[output_pos++] = ' ';
449     }
450 }
451 encoded_output[output_pos] = '\0';
452 }
453
454 void ngram_decode(char *encoded_message, int chunk_size, char
    *decoded_output) {
455     // Removing spaces
456     char clean_encoded[1000];
457     int clean_pos = 0;
458
459     for (int pos = 0; encoded_message[pos] != '\0'; pos++) {
```

Choose Cipher Algorithm:
1. Caesar Cipher
2. Atbash Cipher
3. Augustus Cipher
4. Affine Cipher
5. Vigenère Cipher
6. Gronsfeld Cipher
7. Beaufort Cipher
8. Autokey Cipher
9. Hill Cipher
10. Rail Fence Cipher
11. Ngram Cipher
12. Route Cipher
13. Myszkowski Cipher
Enter the number of the cipher you want to use: 5
Enter the message: laptop
Enter the key for Vigenère Cipher: key
Encrypted: vendsn
Decrypted: laptop

=== Code Execution Successful ===

7. Gronsfeld Cipher:

main.c	Output
<pre>437 } 438 clean_message[clean_pos] = '\0'; 439 440 for (int pos = 0; pos < clean_pos; pos += chunk_size) { 441 int offset = 5; // Fixed shift value for easiness 442 443 for (int j = 0; j < chunk_size; j++) { 444 encoded_output[output_pos++] = ((clean_message[pos + j] 445 - 'A' + offset) % 26) + 'A'; 446 } 447 448 if (pos + chunk_size < clean_pos) { 449 encoded_output[output_pos++] = ' '; 450 } 451 } 452 encoded_output[output_pos] = '\0'; 453 } 454 455 void ngram_decode(char *encoded_message, int chunk_size, char 456 *decoded_output) { 457 // Removing spaces 458 char clean_encoded[1000]; 459 int clean_pos = 0; 460 for (int pos = 0; encoded_message[pos] != '\0'; pos++) {</pre>	<pre>Choose Cipher Algorithm: 1. Caesar Cipher 2. Atbash Cipher 3. Augustus Cipher 4. Affine Cipher 5. Vigenère Cipher 6. Gronsfeld Cipher 7. Beaufort Cipher 8. Autokey Cipher 9. Hill Cipher 10. Rail Fence Cipher 11. Ngram Cipher 12. Route Cipher 13. Myszkowski Cipher Enter the number of the cipher you want to use: 6 Enter the message: sindanika Enter the key for Gronsfeld Cipher (digits only): 1234 Encrypted: tkqhbplob Decrypted: sindanika === Code Execution Successful ===</pre>

8. Beaufort Cipher:

main.c	Output
<pre>437 } 438 clean_message[clean_pos] = '\0'; 439 440 for (int pos = 0; pos < clean_pos; pos += chunk_size) { 441 int offset = 5; // Fixed shift value for easiness 442 443 for (int j = 0; j < chunk_size; j++) { 444 encoded_output[output_pos++] = ((clean_message[pos + j] 445 - 'A' + offset) % 26) + 'A'; 446 } 447 448 if (pos + chunk_size < clean_pos) { 449 encoded_output[output_pos++] = ' '; 450 } 451 } 452 encoded_output[output_pos] = '\0'; 453 } 454 455 void ngram_decode(char *encoded_message, int chunk_size, char 456 *decoded_output) { 457 // Removing spaces 458 char clean_encoded[1000]; 459 int clean_pos = 0; 460 for (int pos = 0; encoded_message[pos] != '\0'; pos++) {</pre>	<pre>Choose Cipher Algorithm: 1. Caesar Cipher 2. Atbash Cipher 3. Augustus Cipher 4. Affine Cipher 5. Vigenère Cipher 6. Gronsfeld Cipher 7. Beaufort Cipher 8. Autokey Cipher 9. Hill Cipher 10. Rail Fence Cipher 11. Ngram Cipher 12. Route Cipher 13. Myszkowski Cipher Enter the number of the cipher you want to use: 7 Enter the message: window screen Enter the key for Beaufort Cipher: cipher Encrypted: gaceqv kgydae Decrypted: window screen === Code Execution Successful ===</pre>

9. Autokey Cipher:

main.c	Output
<pre>437 } 438 clean_message[clean_pos] = '\0'; 439 440 for (int pos = 0; pos < clean_pos; pos += chunk_size) { 441 int offset = 5; // Fixed shift value for easiness 442 443 for (int j = 0; j < chunk_size; j++) { 444 encoded_output[output_pos++] = ((clean_message[pos + j] 445 - 'A' + offset) % 26) + 'A'; 446 } 447 448 if (pos + chunk_size < clean_pos) { 449 encoded_output[output_pos++] = ' '; 450 } 451 452 encoded_output[output_pos] = '\0'; 453 } 454 455 void ngram_decode(char *encoded_message, int chunk_size, char 456 *decoded_output) { 457 // Removing spaces 458 char clean_encoded[1000]; 459 int clean_pos = 0; 460 for (int pos = 0; encoded_message[pos] != '\0'; pos++) {</pre>	<pre>Choose Cipher Algorithm: 1. Caesar Cipher 2. Atbash Cipher 3. Augustus Cipher 4. Affine Cipher 5. Vigenère Cipher 6. Gronsfeld Cipher 7. Beaufort Cipher 8. Autokey Cipher 9. Hill Cipher 10. Rail Fence Cipher 11. Ngram Cipher 12. Route Cipher 13. Myszowski Cipher Enter the number of the cipher you want to use: 8 Enter the message: wireless Enter the key for Autokey Cipher: abc Encrypted: wjtatvwd Decrypted: wireless === Code Execution Successful ===</pre>

10. Hill Cipher:

main.c

Share

Run

Output

Clear

```
437 }
438 clean_message[clean_pos] = '\0';
439
440 for (int pos = 0; pos < clean_pos; pos += chunk_size) {
441     int offset = 5; // Fixed shift value for easiness
442
443     for (int j = 0; j < chunk_size; j++) {
444         encoded_output[output_pos++] = ((clean_message[pos + j]
445             - 'A' + offset) % 26) + 'A';
446     }
447
448     if (pos + chunk_size < clean_pos) {
449         encoded_output[output_pos++] = ' ';
450     }
451 }
452 encoded_output[output_pos] = '\0';
453 }
454
455 void ngram_decode(char *encoded_message, int chunk_size, char
456     *decoded_output) {
457     // Removing spaces
458     char clean_encoded[1000];
459     int clean_pos = 0;
460     for (int pos = 0; encoded_message[pos] != '\0'; pos++) {
```

Choose Cipher Algorithm:
1. Caesar Cipher
2. Atbash Cipher
3. Augustus Cipher
4. Affine Cipher
5. Vigenère Cipher
6. Gronsfeld Cipher
7. Beaufort Cipher
8. Autokey Cipher
9. Hill Cipher
10. Rail Fence Cipher
11. Ngram Cipher
12. Route Cipher
13. Myszkowski Cipher
Enter the number of the cipher you want to use: 9
Enter the message: highlighter
Enter the 4 keys for Hill Cipher as a list (e.g. 5 8 17 3): 3 5 7 6
Encrypted: jtbgvvbgzbkx
Decrypted: highlighterx

=== Code Execution Successful ===

11. Rail Fence Cipher:

main.c

Share

Run

Output

Clear

```
437 }
438 clean_message[clean_pos] = '\0';
439
440 for (int pos = 0; pos < clean_pos; pos += chunk_size) {
441     int offset = 5; // Fixed shift value for easiness
442
443     for (int j = 0; j < chunk_size; j++) {
444         encoded_output[output_pos++] = ((clean_message[pos + j]
445             - 'A' + offset) % 26) + 'A';
446     }
447
448     if (pos + chunk_size < clean_pos) {
449         encoded_output[output_pos++] = ' ';
450     }
451 }
452 encoded_output[output_pos] = '\0';
453 }
454
455 void ngram_decode(char *encoded_message, int chunk_size, char
456     *decoded_output) {
457     // Removing spaces
458     char clean_encoded[1000];
459     int clean_pos = 0;
460     for (int pos = 0; encoded_message[pos] != '\0'; pos++) {
```

Choose Cipher Algorithm:
1. Caesar Cipher
2. Atbash Cipher
3. Augustus Cipher
4. Affine Cipher
5. Vigenère Cipher
6. Gronsfeld Cipher
7. Beaufort Cipher
8. Autokey Cipher
9. Hill Cipher
10. Rail Fence Cipher
11. Ngram Cipher
12. Route Cipher
13. Myszkowski Cipher
Enter the number of the cipher you want to use: 10
Enter the message: I will be there at seven pm
Enter the number of rails for Rail Fence Cipher: 3
Encrypted: Ileeae ilb hr tsvnpw te em
Decrypted: I will be there at seven pm

=== Code Execution Successful ===

12. Ngram Cipher:

main.c	Output
<pre>437 } 438 clean_message[clean_pos] = '\0'; 439 440 for (int pos = 0; pos < clean_pos; pos += chunk_size) { 441 int offset = 5; // Fixed shift value for easiness 442 443 for (int j = 0; j < chunk_size; j++) { 444 encoded_output[output_pos++] = ((clean_message[pos + j] 445 - 'A' + offset) % 26) + 'A'; 446 } 447 448 if (pos + chunk_size < clean_pos) { 449 encoded_output[output_pos++] = ' '; 450 } 451 452 encoded_output[output_pos] = '\0'; 453 } 454 455 void ngram_decode(char *encoded_message, int chunk_size, char 456 *decoded_output) { 457 // Removing spaces 458 char clean_encoded[1000]; 459 int clean_pos = 0; 460 for (int pos = 0; encoded_message[pos] != '\0'; pos++) {</pre>	<pre>Choose Cipher Algorithm: 1. Caesar Cipher 2. Atbash Cipher 3. Augustus Cipher 4. Affine Cipher 5. Vigenère Cipher 6. Gronsfield Cipher 7. Beaufort Cipher 8. Autokey Cipher 9. Hill Cipher 10. Rail Fence Cipher 11. Ngram Cipher 12. Route Cipher 13. Myszkowski Cipher Enter the number of the cipher you want to use: 11 Enter the message: cipher Enter the value of n for Ngram Cipher: 2 Encrypted: HN UM JW Decrypted: CIPHER === Code Execution Successful ===</pre>

13. Route Cipher:

main.c	Output
<pre>437 } 438 clean_message[clean_pos] = '\0'; 439 440 for (int pos = 0; pos < clean_pos; pos += chunk_size) { 441 int offset = 5; // Fixed shift value for easiness 442 443 for (int j = 0; j < chunk_size; j++) { 444 encoded_output[output_pos++] = ((clean_message[pos + j] 445 - 'A' + offset) % 26) + 'A'; 446 } 447 448 if (pos + chunk_size < clean_pos) { 449 encoded_output[output_pos++] = ' '; 450 } 451 452 encoded_output[output_pos] = '\0'; 453 } 454 455 void ngram_decode(char *encoded_message, int chunk_size, char 456 *decoded_output) { 457 // Removing spaces 458 char clean_encoded[1000]; 459 int clean_pos = 0; 460 for (int pos = 0; encoded_message[pos] != '\0'; pos++) {</pre>	<pre>Choose Cipher Algorithm: 1. Caesar Cipher 2. Atbash Cipher 3. Augustus Cipher 4. Affine Cipher 5. Vigenère Cipher 6. Gronsfield Cipher 7. Beaufort Cipher 8. Autokey Cipher 9. Hill Cipher 10. Rail Fence Cipher 11. Ngram Cipher 12. Route Cipher 13. Myszkowski Cipher Enter the number of the cipher you want to use: 12 Enter the message: cellphones Enter the size of the matrix for Route Cipher: 3 Encrypted: cpeehsloxlnx Decrypted: cellphonesxx === Code Execution Successful ===</pre>

14. Myszkowski Cipher:

main.c	Output
<pre>437 } 438 clean_message[clean_pos] = '\0'; 439 440 for (int pos = 0; pos < clean_pos; pos += chunk_size) { 441 int offset = 5; // Fixed shift value for easiness 442 443 for (int j = 0; j < chunk_size; j++) { 444 encoded_output[output_pos++] = ((clean_message[pos + j] 445 - 'A' + offset) % 26) + 'A'; 446 } 447 448 if (pos + chunk_size < clean_pos) { 449 encoded_output[output_pos++] = ' '; 450 } 451 } 452 encoded_output[output_pos] = '\0'; 453 } 454 455 void ngram_decode(char *encoded_message, int chunk_size, char 456 *decoded_output) { 457 // Removing spaces 458 char clean_encoded[1000]; 459 int clean_pos = 0; 460 for (int pos = 0; encoded_message[pos] != '\0'; pos++) {</pre>	<p>Choose Cipher Algorithm:</p> <ol style="list-style-type: none">1. Caesar Cipher2. Atbash Cipher3. Augustus Cipher4. Affine Cipher5. Vigenère Cipher6. Gronsfeld Cipher7. Beaufort Cipher8. Autokey Cipher9. Hill Cipher10. Rail Fence Cipher11. Ngram Cipher12. Route Cipher13. Myszkowski Cipher <p>Enter the number of the cipher you want to use: 13</p> <p>Enter the message: greetings</p> <p>Enter the key for Myszkowski Cipher: mat</p> <p>Encrypted: rtggeneis</p> <p>Decrypted: greetings</p> <p>=== Code Execution Successful ===</p>