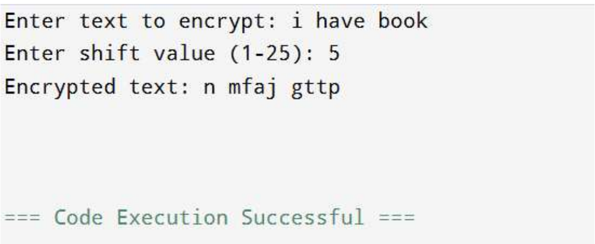
1. Write a C program for Caesar cipher involves replacing each letter of the alphabet with the letter standing k places further down the alphabet, for k in the range 1 through 25.

Ans:
code:

```
#include <stdio.h>
#include <string.h>

void encrypt(char *text, int k) {
    for (int i = 0; text[i] != '\0'; i++) {
        char ch = text[i];
        if (ch >= 'a' && ch <= 'z') {
            ch = (ch - 'a' + k) % 26 + 'a';
        } else if (ch >= 'A' && ch <= 'Z') {
            ch = (ch - 'A' + k) % 26 + 'A';
        }
        text[i] = ch;
    }
}

int main() {
    char text[100];
    int k;

    printf("Enter text to encrypt: ");
    fgets(text, sizeof(text), stdin);
    printf("Enter shift value (1-25): ");
    scanf("%d", &k);

    if (k < 1 || k > 25) {
        printf("Shift value must be between 1 and 25.\n");
        return 1;
    }

    encrypt(text, k);
    printf("Encrypted text: %s\n", text);

    return 0;
}
```

Output:



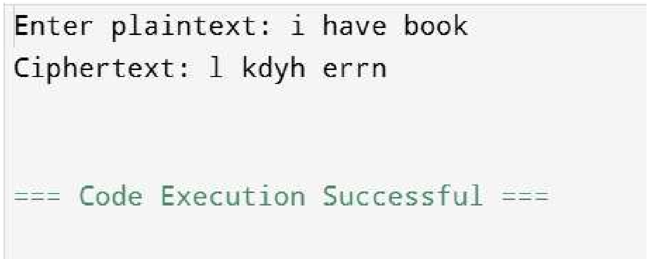
```
Enter text to encrypt: i have book
Enter shift value (1-25): 5
Encrypted text: n mfaj gttp




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

2. Write a C program for monoalphabetic substitution cipher maps a plaintext alphabet to a ciphertext alphabet, so that each letter of the plaintext alphabet maps to a single unique letter of the ciphertext alphabet.

Ans:
Code:

```
#include <stdio.h>
#include <string.h>

void encrypt(char *plaintext, char *ciphertext, char *key) {
    for (int i = 0; plaintext[i] != '\0'; i++) {
        if (plaintext[i] >= 'a' && plaintext[i] <= 'z') {
            ciphertext[i] = key[plaintext[i] - 'a'];
        } else if (plaintext[i] >= 'A' && plaintext[i] <= 'Z') {
            ciphertext[i] = key[plaintext[i] - 'A'] - 32;
        } else {
            ciphertext[i] = plaintext[i];
        }
    }
    ciphertext[strlen(plaintext)] = '\0';
}

int main() {
    char plaintext[100];
    char ciphertext[100];
    char key[26] = {'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'};

    printf("Enter plaintext: ");
    fgets(plaintext, sizeof(plaintext), stdin);
    plaintext[strcspn(plaintext, "\n")] = 0;

    encrypt(plaintext, ciphertext, key);
    printf("Ciphertext: %s\n", ciphertext);

    return 0;
}
```

Output:



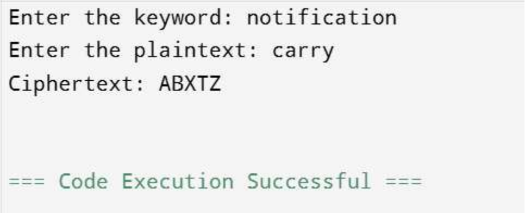
```
Enter plaintext: i have book
Ciphertext: l kdyh errn



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

3. Write a C program for Playfair algorithm is based on the use of a 5 X 5 matrix of letters constructed using a keyword. Plaintext is encrypted two letters at a time using this matrix.

Ans:
Code:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define SIZE 5

void createMatrix(char *key, char matrix[SIZE][SIZE]) {
    int used[26] = {0};
    int k = 0;

    for (int i = 0; key[i] != '\0'; i++) {
        char ch = toupper(key[i]);
        if (ch >= 'A' && ch <= 'Z' && !used[ch - 'A']) {
            used[ch - 'A'] = 1;
            matrix[k / SIZE][k % SIZE] = ch;
            k++;
        }
    }

    for (char ch = 'A'; ch <= 'Z'; ch++) {
        if (ch == 'J') continue;
        if (!used[ch - 'A']) {
            matrix[k / SIZE][k % SIZE] = ch;
            k++;
        }
    }
}

void encrypt(char *plaintext, char matrix[SIZE][SIZE], char *ciphertext) {
    int len = strlen(plaintext);
    for (int i = 0; i < len; i += 2) {
        char a = toupper(plaintext[i]);
        char b = (i + 1 < len) ? toupper(plaintext[i + 1]) : 'X';

        if (a == b) b = 'X';

        int row1, col1, row2, col2;
        for (int r = 0; r < SIZE; r++) {
            for (int c = 0; c < SIZE; c++) {
                if (matrix[r][c] == a) {
                    row1 = r; col1 = c;
                }
                if (matrix[r][c] == b) {
                    row2 = r; col2 = c;
                }
            }
        }

        if (row1 == row2) {
            ciphertext[i] = matrix[row1][(col1 + 1) % SIZE];
            ciphertext[i + 1] = matrix[row2][(col2 + 1) % SIZE];
        } else if (col1 == col2) {
            ciphertext[i] = matrix[(row1 + 1) % SIZE][col1];
            ciphertext[i + 1] = matrix[(row2 + 1) % SIZE][col2];
        } else {
            ciphertext[i] = matrix[row1][col2];
            ciphertext[i + 1] = matrix[row2][col1];
        }
    }
    ciphertext[len] = '\0';
}

int main() {
    char key[100], plaintext[100], ciphertext[100];
    char matrix[SIZE][SIZE];

    printf("Enter the keyword: ");
    scanf("%s", key);
    printf("Enter the plaintext: ");
    scanf("%s", plaintext);

    createMatrix(key, matrix);
    encrypt(plaintext, matrix, ciphertext);

    printf("Ciphertext: %s\n", ciphertext);
    return 0;
}
```

Output:



```
Enter the keyword: notification
Enter the plaintext: carry
Ciphertext: ABXTZ



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

4. As you know, the most frequently occurring letter in English is e. Therefore, the first or second (or perhaps third?) most common character in the message is likely to stand for e. Also, e is often seen in pairs (e.g., meet, fleet, speed, seen, been, agree, etc.). Try to find a character in the ciphertext that decodes to e.

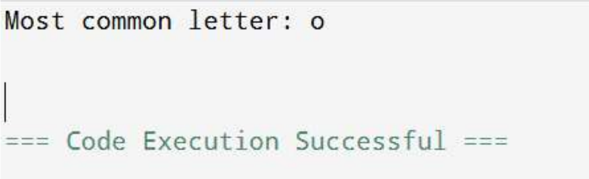
2. The most common word in English is "the." Use this fact to guess the characters that stand for t and h. 3. Decipher the rest of the message by deducing additional words.

Ans:

Code:

```
#include <stdio.h>
#include <string.h>
#define ALPHABET_SIZE 26
void frequencyAnalysis(const char *ciphertext) {
    int frequency[ALPHABET_SIZE] = {0};
    int length = strlen(ciphertext);
    for (int i = 0; i < length; i++) {
        if (ciphertext[i] >= 'a' && ciphertext[i] <= 'z') {
            frequency[ciphertext[i] - 'a']++;
        }
    }
    int maxIndex = 0;
    for (int i = 1; i < ALPHABET_SIZE; i++) {
        if (frequency[i] > frequency[maxIndex]) {
            maxIndex = i;
        }
    }

    printf("Most common letter: %c\n", maxIndex + 'a');
}

int main() {
    const char *ciphertext = "i have book";
    frequencyAnalysis(ciphertext);
    return 0;
}
```

Output:



```
Most common letter: o


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

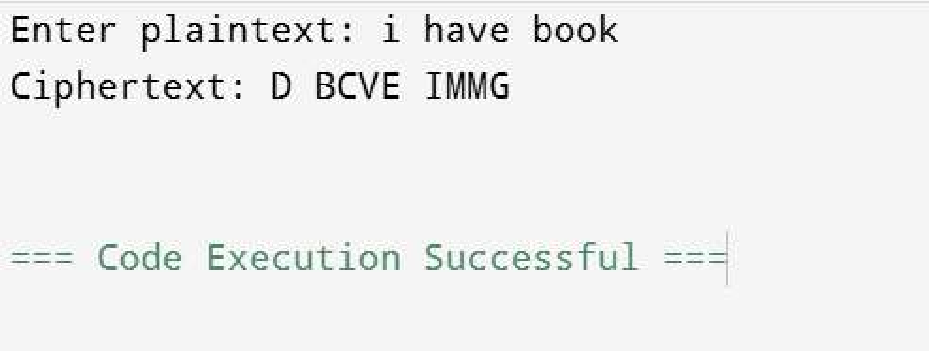
5. Write a C program for monoalphabetic cipher is that both sender and receiver must commit the permuted cipher sequence to memory. A common technique for avoiding this
is to use a keyword from which the cipher sequence can be generated. For example, using
the keyword CIPHER, write out the keyword followed by unused letters in normal order
and match this against the plaintext letters:
plain: 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
cipher: C I P H E R A B D F G J K L M N O Q S T U V W X Y Z

Ans:
Code:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

void generateCipher(char *keyword, char *cipher) {
    int used[26] = {0};
    int index = 0;

    for (int i = 0; keyword[i] != '\0'; i++) {
        char ch = toupper(keyword[i]);
        if (isalpha(ch) && !used[ch - 'A']) {
            cipher[index++] = ch;
            used[ch - 'A'] = 1;
        }
    }
    for (char ch = 'A'; ch <= 'Z'; ch++) {
        if (!used[ch - 'A']) {
            cipher[index++] = ch;
        }
    }
    cipher[index] = '\0';
}

void encrypt(char *plaintext, char *cipher, char *ciphertext) {
    for (int i = 0; plaintext[i] != '\0'; i++) {
        if (isalpha(plaintext[i])) {
            char ch = toupper(plaintext[i]);
            ciphertext[i] = cipher[ch - 'A'];
        } else {
            ciphertext[i] = plaintext[i];
        }
    }
    ciphertext[strlen(plaintext)] = '\0';
}
```

```
int main() {
    char keyword[] = "CIPHER";
    char cipher[27];
    char plaintext[100], ciphertext[100];

    generateCipher(keyword, cipher);

    printf("Enter plaintext: ");
    fgets(plaintext, sizeof(plaintext), stdin);
    plaintext[strcspn(plaintext, "\n")] = 0;

    encrypt(plaintext, cipher, ciphertext);

    printf("Ciphertext: %s\n", ciphertext);
    return 0;
}
```

Output:

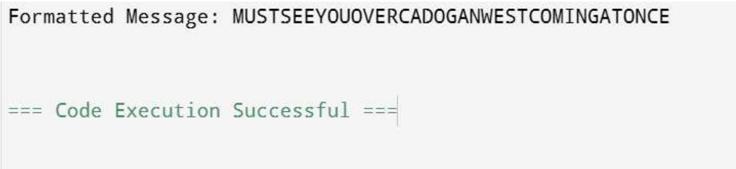


```
Enter plaintext: i have book
Ciphertext: D BCVE IMMG



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

5. Write a C program for Playfair matrix:
MFHI/JK
UNOPQ
ZVWXY
ELARG
DSTBC
Encrypt this message: Must see you over Cadogan West. Coming at once.
Ans:
Code:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#define MATRIX_SIZE 5
char playfairMatrix[MATRIX_SIZE][MATRIX_SIZE] = {
    {'M', 'F', 'H', 'I', 'K'},
    {'U', 'N', 'O', 'P', 'Q'},
    {'Z', 'V', 'W', 'X', 'Y'},
    {'E', 'L', 'A', 'R', 'G'},
    {'D', 'S', 'T', 'B', 'C'}
};
void formatMessage(char *message, char *formatted) {
    int j = 0;
    for (int i = 0; message[i] != '\0'; i++) {
        if (isalpha(message[i])) {
            formatted[j++] = toupper(message[i]);
        }
    }
    formatted[j] = '\0';
}
void encrypt(char *message) {
    char formatted[100];
    formatMessage(message, formatted);
    // Encryption logic will be implemented here
    printf("Formatted Message: %s\n", formatted);
}
int main() {
    char message[] = "Must see you over Cadogan West. Coming at once.";
    encrypt(message);
    return 0;
}
```

Output:



```
Formatted Message: MUSTSEEYOUOVERCADOGANWESTCOMINGATONCE



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

6. Write a C program to Encrypt the message “meet me at the usual place at ten rather than eight oclock” using the Hill cipher with the key.

(9 4)

(5 7)

a. Show your calculations and the result. b. Show the calculations for the corresponding decryption of the ciphertext to recover the original plaintext

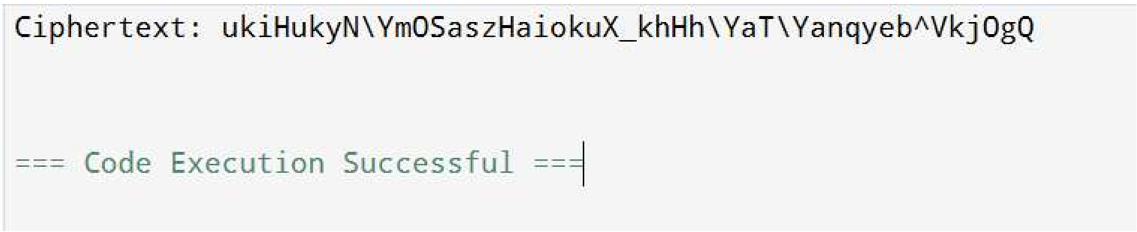
Ans:

Code:

```
#include <stdio.h>
#include <string.h>
#define SIZE 2
void encrypt(char *message, int key[SIZE][SIZE], char *ciphertext) {
  int i, j, k;
  int len = strlen(message);
  int block[SIZE];
  for (i = 0; i < len; i += SIZE) {
    for (j = 0; j < SIZE; j++) {
      block[j] = message[i + j] - 'a';
    }
    for (j = 0; j < SIZE; j++) {
      ciphertext[i + j] = 0;
      for (k = 0; k < SIZE; k++) {
        ciphertext[i + j] += key[j][k] * block[k];
      }
      ciphertext[i + j] = (ciphertext[i + j] % 26) + 'a';
    }
  }
  ciphertext[len] = '\0';
}
```

```
int main() {
    char message[] = "meetmeattheusualplaceattenratherthaneightoclock";
    int key[SIZE][SIZE] = {{9, 4}, {5, 7}};
    char ciphertext[100];
    encrypt(message, key, ciphertext);
    printf("Ciphertext: %s\n", ciphertext);
    return 0;
}
```

Output:



```
Ciphertext: ukiHukyN\YmOSaszHaiokuX_khHh\YaT\Yanqyeb^VkjOgQ


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