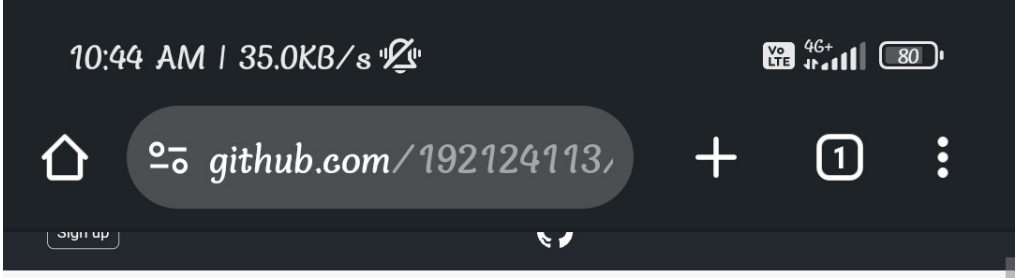
192124113 Add files via upload

Code Blame 29 lines (28 loc) · 958 Bytes

```c
#include <stdio.h>
#include<string.h>
int main() {
    char text[100];
    int shift;
    printf("Enter a string: ");
    fgets(text, sizeof(text), stdin);
    text[strlen(text) - 1] = '\0';
    printf("Enter shift value: ");
    scanf("%d", &shift);
    for (int i = 0; text[i] != '\0'; i++) {
        if (text[i] >= 'A' && text[i] <= 'Z') {
            text[i] = ((text[i] - 'A' + shift) % 26) + 'A';
        } else if (text[i] >= 'a' && text[i] <= 'z') {
            text[i] = ((text[i] - 'a' + shift) % 26) + 'a';
        }
    }
    printf("Encrypted text: %s\n", text);
    for (int i = 0; text[i] != '\0'; i++) {
        if (text[i] >= 'A' && text[i] <= 'Z') {
            text[i] = ((text[i] - 'A' - shift + 26) % 26) + 'A';
        } else if (text[i] >= 'a' && text[i] <= 'z') {
            text[i] = ((text[i] - 'a' - shift + 26) % 26) + 'a';
        }
    }
    printf("Decrypted text: %s\n", text);

    return 0;
}
```




192124113 / CSA5177-CNS Public

Notifications | Fork 0 | Star

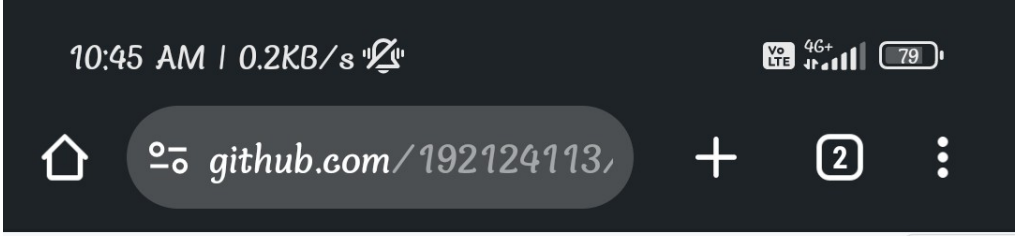
Code | Issues | Pull requests | Actions | Projects | Security | Insights

main | CSA5177-CNS / 2. Playfair.c | Go to file

192124113 Add files via upload — dfb0407 · 3 months ago

Code | Blame | 71 lines (66 loc) · 1.81 KB | Raw

```c
#include <stdio.h>
#include <string.h>
void generateMatrix(char key[], char matrix[5][5]) {
    int i, j, k = 0;
    int len = strlen(key);
    int isPresent[26] = {0};
    for (i = 0; i < len; i++) {
        if (key[i] == 'j') key[i] = 'i';
        if (!isPresent[key[i] - 'a']) {
            matrix[k / 5][k % 5] = key[i];
            isPresent[key[i] - 'a'] = 1;
            k++;
        }
    }
    for (i = 0; i < 26; i++) {
        if (i != ('j' - 'a') && !isPresent[i]) {
            matrix[k / 5][k % 5] = 'a' + i;
            k++;
        }
    }
}
void encrypt(char matrix[5][5], char a, char b) {
    int i, j;
    int row1, col1, row2, col2;
    for (i = 0; i < 5; i++) {
        for (j = 0; j < 5; j++) {
            if (matrix[i][j] == a) {
                row1 = i;
                col1 = j;
            }
            if (matrix[i][j] == b) {
                row2 = i;
                col2 = j;
            }
        }
    }
    if (row1 == row2) {
        col1 = (col1 + 1) % 5;
        col2 = (col2 + 1) % 5;
    } else if (col1 == col2) {
        row1 = (row1 + 1) % 5;
        row2 = (row2 + 1) % 5;
    } else {
        int temp = col1;
        col1 = col2;
        col2 = temp;
    }
    printf("%c%c%c%c", matrix[row1][col1], matrix[row2][col2], matrix[row1][col2], matrix[row2][col1]);
}
int main() {
    char key[25];
    char matrix[5][5];

    printf("Enter the key (up to 25 characters): ");
    scanf("%s", key);

    generateMatrix(key, matrix);

    char plaintext[100];
    printf("Enter the plaintext: ");
    scanf("%s", plaintext);

    int len = strlen(plaintext);
    for (int i = 0; i < len; i += 2) {
        char a = plaintext[i];
        char b = (i + 1 < len) ? plaintext[i + 1] : 'x';
        encrypt(matrix, a, b);
    }

    return 0;
}
```




192124113 / **CSA5177-CNS** Public

Notifi

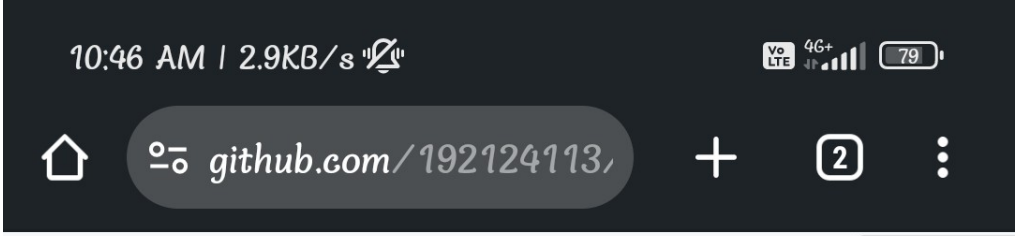
Code | Issues | Pull requests | Actions | Projects | Security | I

main | CSA5177-CNS / **3. Hill cipher.c**

**192124113** Add files via upload

ode | Blame | 29 lines (26 loc) · 869 Bytes

```c
#include <stdio.h>
int main() {
    int key[2][2];
    printf("Enter the 2x2 key matrix:\n");
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            scanf("%d", &key[i][j]);
        }
    }
    while (getchar() != '\n');
    char plaintext[100];
    printf("Enter the plaintext: ");
    fgets(plaintext, sizeof(plaintext), stdin);
    printf("Encrypted text: ");
    for (int i = 0; plaintext[i] != '\0'; i++) {
        if (plaintext[i] >= 'a' && plaintext[i] <= 'z') {
            int x = plaintext[i] - 'a';
            int y = plaintext[i+1] - 'a';

            int encrypted_x = (key[0][0] * x + key[0][1] * y) % 26;
            int encrypted_y = (key[1][0] * x + key[1][1] * y) % 26;

            printf("%c%c", encrypted_x + 'a', encrypted_y + 'a');
            i++;
        }
    }

    return 0;
}
```




192124113 / **CSA5177-CNS** Public

Notifications

Code Issues Pull requests Actions Projects Security Insight

main CSA5177-CNS / **4. Vigenere cipher.c**

**192124113** Add files via upload

ode Blame 46 lines (43 loc) · 1.52 KB

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
void vigenereEncrypt(char *plaintext, char *keyword) {
    int plaintextLen = strlen(plaintext);
    int keywordLen = strlen(keyword);
    for (int i = 0; i < plaintextLen; i++) {
        if (isalpha(plaintext[i])) {
            char key = keyword[i % keywordLen];
            int shift = toupper(key) - 'A';
            if (islower(plaintext[i])) {
                plaintext[i] = (plaintext[i] - 'a' + shift) % 26 + 'a';
            } else {
                plaintext[i] = (plaintext[i] - 'A' + shift) % 26 + 'A';
            }
        }
    }
}
void vigenereDecrypt(char *ciphertext, char *keyword) {
    int ciphertextLen = strlen(ciphertext);
    int keywordLen = strlen(keyword);

    for (int i = 0; i < ciphertextLen; i++) {
        if (isalpha(ciphertext[i])) {
            char key = keyword[i % keywordLen];
            int shift = toupper(key) - 'A';

            if (islower(ciphertext[i])) {
                ciphertext[i] = (ciphertext[i] - 'a' - shift + 26) % 26 + 'a';
            } else {
                ciphertext[i] = (ciphertext[i] - 'A' - shift + 26) % 26 + 'A';
            }
        }
    }
}
int main() {
    char plaintext[] = "HELLOWORLD";
    char keyword[] = "KEY";
    printf("Original Text: %s\n", plaintext);
    vigenereEncrypt(plaintext, keyword);
    printf("Encrypted Text: %s\n", plaintext);
    vigenereDecrypt(plaintext, keyword);
    printf("Decrypted Text: %s\n", plaintext);
    return 0;
}
```

main CSA5177-CNS / 5. Railfence.c

192124113 Add files via upload

Code Blame 53 lines (41 loc) · 1.16 KB

```c
#include <stdio.h>
#include <string.h>
void encryptRailFence(char *message, int rails) {
    int messageLength = strlen(message);

    char railFence[rails][messageLength];

    for (int i = 0; i < rails; i++) {
        for (int j = 0; j < messageLength; j++) {
            railFence[i][j] = ' ';
        }
    }

    int row = 0;
    int direction = 1; // 1 for down, -1 for up

    for (int i = 0; i < messageLength; i++) {
        railFence[row][i] = message[i];

        if (row == 0) {
            direction = 1;
        } else if (row == rails - 1) {
            direction = -1;
        }

        row += direction;
    }

    printf("Encrypted Message: ");
    for (int i = 0; i < rails; i++) {
        for (int j = 0; j < messageLength; j++) {
            if (railFence[i][j] != ' ') {
                printf("%c", railFence[i][j]);
            }
        }
    }
    printf("\n");
}

int main() {
    char message[100];
    int rails;

    printf("Enter the message to encrypt: ");
    gets(message);

    printf("Enter the number of rails: ");
    scanf("%d", &rails);

    encryptRailFence(message, rails);

    return 0;
}
```

Code | Issues | Pull requests | Actions | Projects | Security

main | CSA5177-CNS / 6. DES.c

192124113 Add files via upload

de | Blame | 28 lines (20 loc) · 653 Bytes

```c
#include <stdio.h>
#include <stdint.h>
static int initial_permutation[64] = {
};

void initialPermutation(uint64_t* data) {
}

void desRound(uint32_t* left, uint32_t* right, uint32_t subkey) {
}

void desEncrypt(uint64_t* data, uint64_t* key) {
}

int main() {
    uint64_t plaintext = 0x0123456789ABCDEF; // 64-bit plaintext
    uint64_t key = 0x133457799BBCDFF1;        // 64-bit key

    printf("Original Plaintext: 0x%016llX\n", plaintext);
    printf("Key: 0x%016llX\n", key);

    initialPermutation(&plaintext);
    desEncrypt(&plaintext, &key);

    printf("Encrypted Text: 0x%016llX\n", plaintext);

    return 0;
}
```

192124113 Add files via upload dfb0

Code Blame 70 lines (50 loc) · 1.59 KB

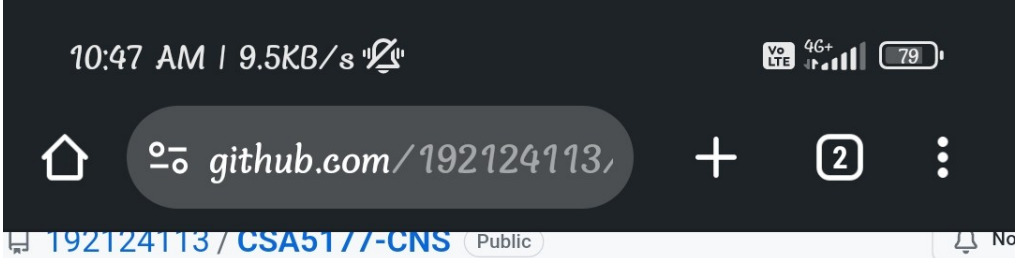
```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int isPrime(int num) {
    if (num <= 1) return 0;
    if (num <= 3) return 1;

    if (num % 2 == 0 || num % 3 == 0) return 0;

    for (int i = 5; i * i <= num; i += 6) {
        if (num % i == 0 || num % (i + 2) == 0) return 0;
    }

    return 1;
}

int gcd(int a, int b) {
    if (b == 0) return a;
    return gcd(b, a % b);
}

void generateKeys(int p, int q, int *n, int *e, int *d) {
    *n = p * q;
    int phi = (p - 1) * (q - 1);

    do {
        printf("Enter a value for 'e' (must be coprime with phi): ");
        scanf("%d", e);
    } while (gcd(*e, phi) != 1);

    for (*d = 1; (*d * *e) % phi != 1; (*d)++);
}

int rsaEncrypt(int message, int e, int n) {
    return (int)pow(message, e) % n;
}

int rsaDecrypt(int encryptedMessage, int d, int n) {
    return (int)pow(encryptedMessage, d) % n;
}

int main() {
    int p, q, n, e, d;

    printf("Enter two prime numbers 'p' and 'q': ");
    scanf("%d %d", &p, &q);

    if (!isPrime(p) || !isPrime(q)) {
        printf("p and q must be prime numbers.\n");
        return 1;
    }

    generateKeys(p, q, &n, &e, &d);

    int message;
    printf("Enter the message to be encrypted: ");
    scanf("%d", &message);

    printf("Original Message: %d\n", message);

    int encryptedMessage = rsaEncrypt(message, e, n);
    printf("Encrypted Message: %d\n", encryptedMessage);

    int decryptedMessage = rsaDecrypt(encryptedMessage, d, n);
    printf("Decrypted Message: %d\n", decryptedMessage);

    return 0;
}
```

```c
#include<stdio.h>
long int power(int a,int b,int mod)
 {
  long long int t;
  if(b==1)
  return a;
  t=power(a,b/2,mod);
  if(b%2==0)
   return (t*t)%mod;
  else
   return (((t*t)%mod)*a)%mod;
 }
 long long int calculateKey(int a,int x,int n)
 {
  return power(a,x,n);
 }
 int main()
 {
  int n,g,x,a,y,b;
  printf("Enter the value of n and g : ");
  scanf("%d%d",&n,&g);
  printf("Enter the value of x for the first person : ");
  scanf("%d",&x);  a=power(g,x,n);
  printf("Enter the value of y for the second person : ");
  scanf("%d",&y);  b=power(g,y,n);
  printf("key for the first person is : %lld\n",power(b,x,n));
  printf("key for the second person is : %lld\n",power(a,y,n));
  return 0;
 }
```

192124113 Add files via upload

Code Blame 25 lines (24 loc) · 696 Bytes

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_INPUT_LENGTH 100
void compute_md5(const char *input, unsigned char *md5_hash) {
    MD5_CTX ctx;
    MD5_Init(&ctx);
    MD5_Update(&ctx, input, strlen(input));
    MD5_Final(md5_hash, &ctx);
}
int main() {
    char input[MAX_INPUT_LENGTH];
    unsigned char md5_hash[MD5_DIGEST_LENGTH];
    printf("Enter the input string: ");
    fgets(input, MAX_INPUT_LENGTH, stdin);
    input[strcspn(input, "\n")] = '\0';
    compute_md5(input, md5_hash);
    printf("MD5 Hash: ");
    for (int i = 0; i < MD5_DIGEST_LENGTH; i++) {
        printf("%02x", md5_hash[i]);
    }
    printf("\n");
    return 0;
}

```




192124113 / CSA5177-CNS Public

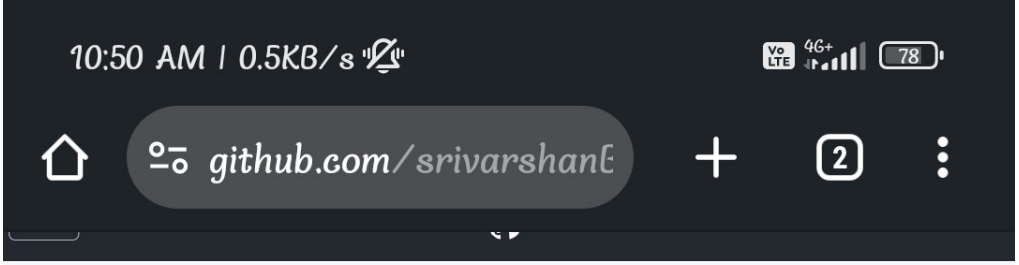
Code Issues Pull requests Actions Projects Security

main CSA5177-CNS / 10. SHA1.c

192124113 Add files via upload

Code Blame 25 lines (22 loc) · 608 Bytes

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <openssl/sha.h>
void sha1_hash(const char *input, unsigned char *hash) {
    SHA_CTX context;
    SHA1_Init(&context);
    SHA1_Update(&context, input, strlen(input));
    SHA1_Final(hash, &context);
}
int main() {
    const char *input = "Hello, World!";
    unsigned char hash[SHA_DIGEST_LENGTH];

    sha1_hash(input, hash);

    printf("Input: %s\n", input);
    printf("SHA-1 Hash: ");
    for (int i = 0; i < SHA_DIGEST_LENGTH; i++) {
        printf("%02x", hash[i]);
    }
    printf("\n");

    return 0;
}
```




srivarshanB / CSA-5116-cryptography Public

Notifications | Fork 0 | Star 0

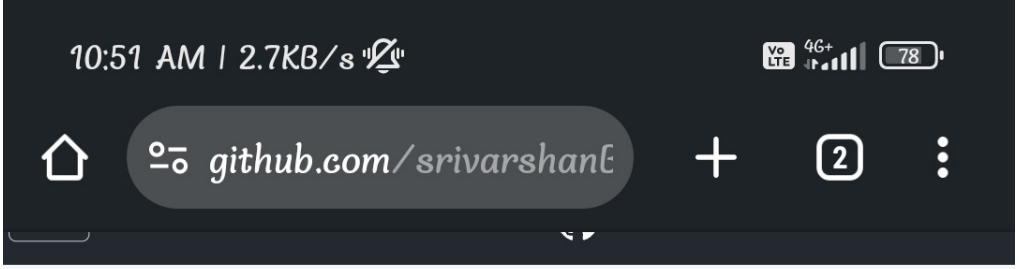
Code | Issues | Pull requests | Actions | Projects | Security | Insights

main | CSA-5116-cryptography / mono-alphabetic cipher exp4.c | Go to file

srivarshanB Add files via upload — 6774377 · 4 days ago | History

Code | Blame | 23 lines (22 loc) · 531 Bytes | Raw

```c
#include<stdio.h>
int main(){
        char alpha[100]="abcdefghijklmnopqrstuvwxyz",key[100]="zyxwvutsrqponmlkjihgfedcba",plain[100],cipher[100];
        int m=0,index[100],i,j;
        printf("Enter plain text :");
        scanf("%s",&plain);
        for(i=0;i<strlen(plain);i++){
                for(j=0;j<strlen(alpha);j++){
                        if(plain[i]==alpha[j]){
                                index[m]=j;
                                m++;
                        }
                }
        }
        printf("Cipher text: ");
        for(i=0;i<strlen(plain);i++){
                cipher[i]=key[index[i]];
                printf("%c",cipher[i]);
        }
        printf("\n Plain text  %s ",plain);
        return 0;
}

```




srivarshanB / CSA-5116-cryptography Public

Notifications Fork 0

Code Issues Pull requests Actions Projects Security Insights

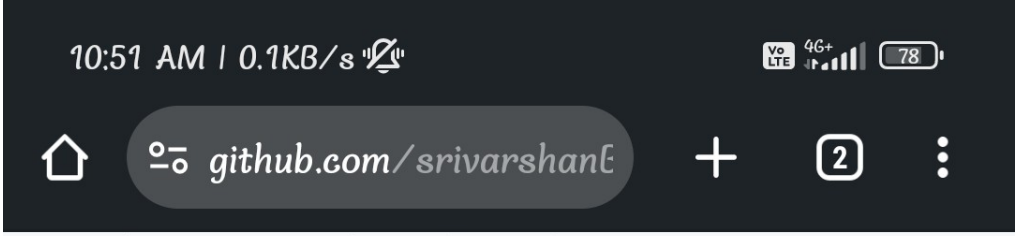
main CSA-5116-cryptography / 11.Polyalphabetic cipher.cpp

Go to file

srivarshanB Add files via upload 1c0e566 · 2 hours

Code Blame 42 lines (37 loc) · 823 Bytes Raw

```
#include<stdio.h>
#include<string.h>
int main()
{
    char pt[20]={'\0'},ct[20]={'\0'},key[20]={'\0'},rt[20]={'\0'};
    int i,j;

    printf("\n enter the plain text:");
    scanf("%s",pt);
    printf("\n enter the key:");
    scanf("%s",key);
    j=0;
    for(i=strlen(key);i<strlen(pt);i++)
    {
    if(j==strlen(key))
    {
    j=0;
    }
    key[i]=key[j];
    j++;
    }
    printf("\n new key is:%s",key);


    for(i=0;i<strlen(pt);i++)
    {
        ct[i]=(((pt[i]-97)+(key[i]-97))%26)+97;
    }
    printf("\n \n cipher text is:%s",ct);

    for(i=0;i<strlen(ct);i++)
    {
    if(ct[i]<key[i])
    {
        rt[i]=26+((ct[i]-97)-(key[i]-97))+97;
    }
    else
        rt[i]=(((ct[i]-97)-(key[i]-97))%26)+97;
    }
    printf("\n \n plain text is:%s",rt);

}
```




Code Issues Pull requests Actions Projects Security Insights

main CSA-5116-cryptography / 12. Digital signature.cpp Go to file

srivarshanB Add files via upload 1c0e566 · 2 hours ago History

Code Blame 61 lines (45 loc) · 1.61 KB Raw

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define PRIME_NUMBER 11

unsigned int public_key;

void generate_keys(unsigned int *private_key, unsigned int *public_key) {
  *private_key = rand() % PRIME_NUMBER;


  *public_key = (unsigned int)pow(*private_key, 2) % PRIME_NUMBER;
}

unsigned int mod_inverse(unsigned int a, unsigned int m) {

  for (unsigned int i = 1; i < m; i++)
    if ((a * i) % m == 1)
      return i;
  return 0;
}

unsigned int sign(unsigned int message, unsigned int private_key) {

  unsigned int k = rand() % PRIME_NUMBER;
  unsigned int r = (k * public_key) % PRIME_NUMBER;
  unsigned int s = (mod_inverse(k, PRIME_NUMBER) * (message + private_key * r)) % PRIME_NUMBER;

  return (r << 16) | s;
}

int verify(unsigned int message, unsigned int signature, unsigned int public_key) {
  unsigned int r = signature >> 16;
  unsigned int s = signature & 0xffff;

  unsigned int w = mod_inverse(s, PRIME_NUMBER);

  unsigned int u1 = (message * w) % PRIME_NUMBER;
  unsigned int u2 = (r * w) % PRIME_NUMBER;
  unsigned int v = ((unsigned int)pow(public_key, u1) * (unsigned int)pow(r, u2)) % PRIME_NUMBER;
  return v == r;
}

int main() {
  unsigned int private_key;
  generate_keys(&private_key, &public_key);
  unsigned int message;
  printf("Enter the message to be signed: ");
  scanf("%u", &message);
  unsigned int signature = sign(message, private_key);
  int is_valid = verify(message, signature, public_key);
  if (is_valid) {
    printf("The signature is valid.\n");
  } else {
    printf("The signature is invalid.\n");
  }

  return 0;
}
```

```c
#include<stdio.h>
#include<string.h>
void encrypt(char message[],int key){
        int len=strlen(message),row=(len+key-1)/key,m=0;
        char encry[100][100];
        int index=0;
        for(int i=0;i<row;i++){
                for(int j=0;j<key;j++){
                        if(m<len){
                                encry[i][j]=message[m];
                                m++;
                        }
                        else{
                                encry[i][j]='X';
                        }

                }
        }
        for(int j=0;j<key;j++){
                for(int i=0;i<row;i++){
                        if(encry[i][j]!='X')
                                printf("%c ",encry[i][j]);
                }
        }
}
int main(){
        char message[100];
        int key;
        printf("Enter the message:");
        scanf("%s",&message);
        printf("\nEnter the key: ");
        scanf("%d",&key);
        printf("Encryted message is :\n");
        encrypt(message,key);
}
```