**PASCALL**

**OUTPUT**

**ENCRYPT**

Enter the message: Hello World

Enter the cipher (or max): 5

Enter 1 to encrypt, 2 to decrypt, and 3 to solve: 1

Original message: HELLO WORLD

Encrypted message: MJQQT BTWQI

Press <Enter> to quit

**DECRYPT**

Enter the message: MJQQT BTWQI

Enter the cipher (or max): 5

Enter 1 to encrypt, 2 to decrypt, and 3 to solve: 2

Original message: MJQQT BTWQI

Decrypted message: HELLO WORLD

Press <Enter> to quit

**SOLVE**

Enter the message: HAL

Enter the cipher (or max): 26

Enter 1 to encrypt, 2 to decrypt, and 3 to solve: 3

Cipher: 26

Decrypted message: HAL

Cipher: 25

Decrypted message: GZK

Cipher: 24

Decrypted message: FYJ

Cipher: 23

Decrypted message: EXI

Cipher: 22

Decrypted message: DWH

Cipher: 21

Decrypted message: CVG

Cipher: 20

Decrypted message: BUF

Cipher: 19

Decrypted message: ATE

Cipher: 18

Decrypted message: ZSD

Cipher: 17

Decrypted message: YRC

Cipher: 16

Decrypted message: XQB

Cipher: 15

Decrypted message: WPA

Cipher: 14

Decrypted message: VOZ

Cipher: 13

Decrypted message: UNY

Cipher: 12

Decrypted message: TMX

Cipher: 11

Decrypted message: SLW

Cipher: 10

Decrypted message: RKV

Cipher: 9

Decrypted message: QJU

Cipher: 8

Decrypted message: PIT

Cipher: 7

Decrypted message: OHS

Cipher: 6

Decrypted message: NGR

Cipher: 5

Decrypted message: MFQ

Cipher: 4

Decrypted message: LEP

Cipher: 3

Decrypted message: KDO

Cipher: 2

Decrypted message: JCN

Cipher: 1

Decrypted message: IBM

Cipher: 0

Decrypted message: HAL

Press <Enter> to quit

**SOURCE**

program caesarcipher;

{$mode objfpc}{$H+}

uses sysutils,

{$IFDEF UNIX}{$IFDEF UseCThreads}

cthreads,

{$ENDIF}{$ENDIF}

Classes

{ you can add units after this };

{Anthony Barranco Caesar Cipher Program in Pascal}

{Variable declaration}

var

cipher: integer;

userCase: integer;

input: string;

{Encrypts the input}

procedure encrypt(var input: string; cipher: integer);

var

i: integer;

output: string;

begin

output := input; {Must equal the length of the input, or won't allocate space properly}

for i := 1 to length(input) do

case input[i] of

'A'..'Z': output[i] := chr(ord('A') + (ord(input[i]) - ord('A') + cipher) mod 26);

end;

writeln ('Encrypted message: ', output);

end;

{Decrypts the input}

procedure decrypt(var input: string; cipher: integer);

var

i: integer;

output: string;

begin

output := input; {Must equal the length of the input, or won't allocate space properly}

for i := 1 to length(input) do

case input[i] of

'A'..'Z': output[i] := chr(ord('A') + (ord(input[i]) - ord('A') - cipher + 26) mod 26);

end;

writeln ('Decrypted message: ', output);

end;

{Solves the input}

procedure solve(var input: string; cipher: integer);

var

i: integer;

Begin

for i := cipher downto 0 do

begin

writeln('Cipher: ', i);

decrypt(input, (i - (28 \* i)));

end;

end;

begin

{cipher := 3; }

{input := 'Manual message';}

write('Enter the message: ');

readln(input);

input := UpperCase(input);{Convert input to upper case}

write('Enter the cipher (or max): ');

read(cipher);

write('Enter 1 to encrypt, 2 to decrypt, and 3 to solve: ');

read(userCase);

writeln();

{perform the user's input}

case userCase of

1 : Begin

writeln ('Original message: ', input);

encrypt(input, cipher);

end; {end e}

2 : Begin

writeln ('Original message: ', input);

decrypt(input, cipher);

end; {end d}

3 : Begin

solve(input, cipher);

end;

end; {End case}

writeln();

writeln('Press <Enter> to quit');

readln;

readln;

end.