**ML**

**OUTPUT**

Standard ML of New Jersey v110.75 [built: Sat Sep 29 12:51:13 2012]

- use "caesarcipher.sml";

[opening caesarcipher.sml]

[autoloading]

[library $SMLNJ-BASIS/basis.cm is stable]

[autoloading done]

val convertN = fn : char -> int

val convertCapN = fn : char -> int

val convertA = fn : int -> char

val convertCapA = fn : int -> char

val shift = fn : char \* int -> char

val seperate = fn : char list \* int -> char list

val encrypt = fn : string \* int -> string

val decrypt = fn : string \* int -> string

val solve = fn : string \* int -> unit

val it = () : unit

- encrypt("Hello World",5);

val it = "Mjqqt Btwqi" : string

- decrypt("MjqqtBtwqi",5);

val it = "Hello World" : string

- solve("HAL", 26);

HAL

GZK

FYJ

EXI

DWH

CVG

BUF

ATE

ZSD

YRC

XQB

WPA

VOZ

UNY

TMX

SLW

RKV

QJU

PIT

OHS

NGR

MFQ

LEP

KDO

JCN

IBM

val it = () : unit

-

**SOURCE**

fun convertN(a) = ord a - ord #"a"

fun convertCapN(a) = ord a - ord#"A"

fun convertA(n) = chr(ord #"a" + n)

fun convertCapA(n) = chr(ord #"A" + n)

fun shift (a,n) = if Char.isLower a then convertA((convertN(a) + n) mod 26)

else convertCapA((convertCapN(a) + n) mod 26)

fun seperate(nil,n) = nil

| seperate(x::xs,n) = shift(x,n) :: seperate(xs,n)

fun encrypt(a,n) = implode(seperate(explode(a),n))

fun decrypt(a,n) = encrypt(a,~n)

fun solve(a,max) =

let

fun findrange(n) = List.tabulate(n, fn x => x);

val maximum = findrange(max)

fun attemptToSolve(a,nil) = nil

| attemptToSolve(a,x::xs) = decrypt(a,x) :: attemptToSolve(a,xs);

fun printSolve(nil) = ()

| printSolve(x::xs) =

(

print(x);

print("\n");

printSolve(xs)

);

in

printSolve(attemptToSolve(a,maximum))

end;