Programming In The Past

Colin MacDonald

March 26, 2021

1 Caesar Cipher

1.1 Fortran

1.1.1 Code

```
!Colin MacDonald
!Run this online: https://www.jdoodle.com/ia/bkM
program CaesarCipher
    implicit none

    character(99) :: str = "This is a test string from Alan"
    integer, parameter :: shiftAmount = 8

    call encrypt(str, shiftAmount)
    write(*, *) str
    call decrypt(str, shiftAmount)
    write(*, *) str

    str="As with all single-alphabet substitution ciphers, the Caesar cipher is easily broken."
    call encrypt(str, 19)
    write(*, *) str
    call decrypt(str, 19)
    write(*, *) str

    str="HAL"
    call solve(str, 26)

    str="wb qfmdhcufodvm, o qosgof qwdvsf, ozgc ybckb og qosgofg qwdvsf, hvs gvwth qwdvsf, qosgofg
        ..."
    call solve(str, 14)
    contains

    subroutine encrypt(str, shiftAmount)
        character(*), intent(inout) :: str
        integer, intent(in) :: shiftAmount
        integer :: i
```

```
        character :: chr
        integer :: asciCode
        character :: newChar
        character(99) :: result

        do i = 1, len(str)
            chr=str(i:i)
            asciCode=iachar(chr)
            if(asciCode >= 65 .and. asciCode<=91) then
                result=result(1:i-1)//achar(modulo(asciCode - 65 + shiftAmount, 26) + 65)
            else if(asciCode>=97 .and. asciCode<=122) then
                result=result(1:i-1)//achar(modulo(asciCode - 97 + shiftAmount, 26) + 97)
            else
                result=result(1:i-1)//chr
            endif
        end do
        str=result

    end subroutine encrypt

    subroutine decrypt(str, shiftAmount)
        character(*), intent(inout) :: str
        integer, intent(in) :: shiftAmount
        integer shiftInvert

        shiftInvert = 26 - shiftAmount
        call encrypt(str, shiftInvert)
    end subroutine decrypt

    subroutine solve(str, maxShiftValue)
        character(*), intent(inout) :: str
        integer, intent(in) :: maxShiftValue
        character(99) :: originalStr
        integer i
        originalStr = str
        i = maxShiftValue
        do while (i >= 0)
            str = originalStr
            call encrypt(str, i)
            write(*, *) "Caesar ", i, ": ", str
            i=i-1
        end do
    end subroutine solve

end program CaesarCipher
```

1.1.2 Output

```
Bpqa qa i bmab abzqvo nzwu Itiv
This is a test string from Alan
Tl pbma tee lbgzex-teiatuxm lnulmbmnmbhg vbiaxkl, max Vtxltk vbiaxk bl xtlber ukhdxg.
As with all single-alphabet substitution ciphers, the Caesar cipher is easily broken.
Caesar          26 : HAL
Caesar          25 : GZK
```

```
Caesar         24 : FYJ
Caesar         23 : EXI
Caesar         22 : DWH
Caesar         21 : CVG
Caesar         20 : BUF
Caesar         19 : ATE
Caesar         18 : ZSD
Caesar         17 : YRC
Caesar         16 : XQB
Caesar         15 : WPA
Caesar         14 : VOZ
Caesar         13 : UNY
Caesar         12 : TMX
Caesar         11 : SLW
Caesar         10 : RKV
Caesar          9 : QJU
Caesar          8 : PIT
Caesar          7 : OHS
Caesar          6 : NGR
Caesar          5 : MFQ
Caesar          4 : LEP
Caesar          3 : KDO
Caesar          2 : JCN
Caesar          1 : IBM
Caesar          0 : HAL
Caesar         14 : kp etarvqitcrja, c ecguct ekrjgt, cnuq mpqyp cu ecguctu ekrjgt, vjg ujkhv
    ekrjgt, ecguctu ...
Caesar         13 : jo dszquphsbqiz, b dbftbs djqifs, bmtp lopxo bt dbftbst djqifs, uif tijgu
    djqifs, dbftbst ...
Caesar         12 : in cryptography, a caesar cipher, also known as caesars cipher, the shift
    cipher, caesars ...
Caesar         11 : hm bqxosnfqzogx, z bzdrzq bhogdq, zkrn jmnvm zr bzdrzqr bhogdq, sgd rghes
    bhogdq, bzdrzqr ...
Caesar         10 : gl apwnrmepynfw, y aycqyp agnfcp, yjqm ilmul yq aycqypq agnfcp, rfc qfgdr
    agnfcp, aycqypq ...
Caesar          9 : fk zovmqldoxmev, x zxbpxo zfmebo, xipl hkltk xp zxbpxop zfmebo, qeb pefcq
    zfmebo, zxbpxop ...
Caesar          8 : ej ynulpkcnwldu, w ywaown yeldan, whok gjksj wo ywaowno yeldan, pda odebp
    yeldan, ywaowno ...
Caesar          7 : di xmtkojbmvkct, v xvznvm xdkczm, vgnj fijri vn xvznvmn xdkczm, ocz ncdao
    xdkczm, xvznvmn ...
Caesar          6 : ch wlsjnialujbs, u wuymul wcjbyl, ufmi ehiqh um wuymulm wcjbyl, nby mbczn
    wcjbyl, wuymulm ...
Caesar          5 : bg vkrimhzktiar, t vtxltk vbiaxk, telh dghpg tl vtxltkl vbiaxk, max labym
    vbiaxk, vtxltkl ...
Caesar          4 : af ujqhlgyjshzq, s uswksj uahzwj, sdkg cfgof sk uswksjk uahzwj, lzw kzaxl
    uahzwj, uswksjk ...
Caesar          3 : ze tipgkfxirgyp, r trvjri tzgyvi, rcjf befne rj trvjrij tzgyvi, kyv jyzwk
    tzgyvi, trvjrij ...
Caesar          2 : yd shofjewhqfxo, q squiqh syfxuh, qbie ademd qi squiqhi syfxuh, jxu ixyvj
    syfxuh, squiqhi ...
Caesar          1 : xc rgneidvgpewn, p rpthpg rxewtg, pahd zcdlc ph rpthpgh rxewtg, iwt hwxui
    rxewtg, rpthpgh ...
Caesar          0 : wb qfmdhcufodvm, o qosgof qwdvsf, ozgc ybckb og qosgofg qwdvsf, hvs gvwth
    qwdvsf, qosgofg ...
```

1.2 COBOL

1.2.1 Code

```
*>Colin MacDonald
*> Run this online: https://www.jdoodle.com/ia/bku
IDENTIFICATION DIVISION.
PROGRAM-ID. CAESAR-CIPHER.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    FUNCTION ENCRYPT
    FUNCTION DECRYPT
    FUNCTION SOLVE
    .
DATA DIVISION.
    WORKING-STORAGE SECTION.
        01 theStr PIC X(99) value "This is a test string from Alan".
        01 shiftAmount PIC 99 value 8.
        01 res PIC X(99).

PROCEDURE DIVISION.

    MOVE FUNCTION ENCRYPT(theStr, shiftAmount) TO res.
    display res.

    MOVE FUNCTION DECRYPT(res, shiftAmount) TO res.
    display res.

    move "As with all single-alphabet substitution ciphers, the Caesar cipher is easily broken." to
        theStr
    MOVE FUNCTION ENCRYPT(theStr, 19) TO res.
    display res.

    MOVE FUNCTION DECRYPT(res, 19) TO res.
    display res.

    move "HAL" to theStr
    move FUNCTION SOLVE(theStr , 26) to res.

    move "wb qfmdhcufodvm, o qosgof qwdvsf, ozgc ybckb og qosgofg qwdvsf, hvs gvwth qwdvsf, qosgofg
        ..." to theStr.
    move FUNCTION solve(theStr, 14) to res.

    STOP RUN.
    END PROGRAM CAESAR-CIPHER.



IDENTIFICATION DIVISION.
FUNCTION-ID. ENCRYPT.
DATA DIVISION.
LOCAL-STORAGE SECTION.
    01 chr PIC X.
    01 chrCode PIC 999.
    01 newChr PIC X.
    01 counter PIC 99 value 1.
```

```
01 resultLengthTemp PIC 99 value 1.
01 skip1 PIC 9 value 0.
01 skip2 PIC 9 value 0.


LINKAGE SECTION.
01 str PIC X(99).
01 shift PIC 99.
01 result PIC X(99).


PROCEDURE DIVISION USING str, shift RETURNING result.
*>display str
PERFORM A-PARA UNTIL counter = function length(str).


A-PARA.
    move str(counter:1) to chr.
    compute chrCode = FUNCTION ORD(chr) - 1.


    *>DISPLAY FUNCTION CHAR(chrCode) WITH NO ADVANCING.
    if (chrCode >= 65 and chrCode <= 91) then
       move function char(function mod(chrCode + shift - 65, 26) + 66) to newChr
     *>move FUNCTION CONCATENATE(result, newChr) to result
     STRING newChr DELIMITED BY spaces
     INTO result
     WITH POINTER resultLengthTemp
     END-STRING
  else
      move 1 to skip1
  END-IF.

  if (chrCode >= 97 and chrCode <= 122) then
     move function char(function mod(chrCode + shift - 97, 26) + 98) to newChr
     STRING newChr DELIMITED BY spaces
     INTO result
     WITH POINTER resultLengthTemp
     END-STRING
  else
      move 1 to skip2
  END-IF.

  if skip1 is equal to 1 and skip2 is equal to 1 then
     STRING chr DELIMITED BY size
     INTO result
     WITH POINTER resultLengthTemp
     END-STRING
  END-IF.

  move 0 to skip1
  move 0 to skip2
    add 1 to counter.

END FUNCTION ENCRYPT.
```

```
IDENTIFICATION DIVISION.
FUNCTION-ID. DECRYPT.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    FUNCTION encrypt
    .
DATA DIVISION.
WORKING-STORAGE SECTION.
    01 shiftInvert PIC 99.
LINKAGE SECTION.
    01 str  PIC X(99).
    01 shift PIC 99.
    01 result PIC X(99).

PROCEDURE DIVISION USING str, shift RETURNING result.
    SUBTRACT shift from 26 GIVING shiftInvert.
   MOVE FUNCTION ENCRYPT(str, shiftInvert) TO result.

   END FUNCTION DECRYPT.


IDENTIFICATION DIVISION.
FUNCTION-ID. SOLVE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    FUNCTION encrypt
    .
DATA DIVISION.
WORKING-STORAGE SECTION.
    01 myCount PIC 99.
LINKAGE SECTION.
    01 str PIC X(99).
    01 maxShiftValue PIC 99.
    01 result PIC X(99).

PROCEDURE DIVISION USING str, maxShiftValue RETURNING result.
    set myCount to maxShiftValue.
    PERFORM loop UNTIL myCount = 0.

    loop.
        move FUNCTION ENCRYPT(str, myCount) to result
        *>display result
        display "Caesar ", myCount, ": ", result
        subtract 1 from myCount.


   END FUNCTION SOLVE.
```

1.2.2 Output

```
Bpqa qa i bmab abzqvo nzwu Itiv
This is a test string from Alan
Tl pbma tee lbgzex-teiatuxm lnulmbmnmbhg vbiaxkl, max Vtxltk vbiaxk bl xtlber ukhdxg.
As with all single-alphabet substitution ciphers, the Caesar cipher is easily broken.
Caesar 26: HAL
Caesar 25: GZK
Caesar 24: FYJ
Caesar 23: EXI
Caesar 22: DWH
Caesar 21: CVG
Caesar 20: BUF
Caesar 19: ATE
Caesar 18: ZSD
Caesar 17: YRC
Caesar 16: XQB
Caesar 15: WPA
Caesar 14: VOZ
Caesar 13: UNY
Caesar 12: TMX
Caesar 11: SLW
Caesar 10: RKV
Caesar 09: QJU
Caesar 08: PIT
Caesar 07: OHS
Caesar 06: NGR
Caesar 05: MFQ
Caesar 04: LEP
Caesar 03: KDO
Caesar 02: JCN
Caesar 01: IBM
Caesar 00: HAL
Caesar 14: kp etarvqitcrja, c ecguct ekrjgt, cnuq mpqyp cu ecguctu ekrjgt, vjg ujkhv ekrjgt,
    ecguctu ...
Caesar 13: jo dszquphsbqiz, b dbftbs djqifs, bmtp lopxo bt dbftbst djqifs, uif tijgu djqifs,
    dbftbst ...
Caesar 12: in cryptography, a caesar cipher, also known as caesars cipher, the shift cipher,
    caesars ...
Caesar 11: hm bqxosnfqzogx, z bzdrzq bhogdq, zkrn jmnvm zr bzdrzqr bhogdq, sgd rghes bhogdq,
    bzdrzqr ...
Caesar 10: gl apwnrmepynfw, y aycqyp agnfcp, yjqm ilmul yq aycqypq agnfcp, rfc qfgdr agnfcp,
    aycqypq ...
Caesar 09: fk zovmqldoxmev, x zxbpxo zfmebo, xipl hkltk xp zxbpxop zfmebo, qeb pefcq zfmebo,
    zxbpxop ...
Caesar 08: ej ynulpkcnwldu, w ywaown yeldan, whok gjksj wo ywaowno yeldan, pda odebp yeldan,
    ywaowno ...
Caesar 07: di xmtkojbmvkct, v xvznvm xdkczm, vgnj fijri vn xvznvmn xdkczm, ocz ncdao xdkczm,
    xvznvmn ...
Caesar 06: ch wlsjnialujbs, u wuymul wcjbyl, ufmi ehiqh um wuymulm wcjbyl, nby mbczn wcjbyl,
    wuymulm ...
Caesar 05: bg vkrimhzktiar, t vtxltk vbiaxk, telh dghpg tl vtxltkl vbiaxk, max labym vbiaxk,
    vtxltkl ...
Caesar 04: af ujqhlgyjshzq, s uswksj uahzwj, sdkg cfgof sk uswksjk uahzwj, lzw kzaxl uahzwj,
    uswksjk ...
Caesar 03: ze tipgkfxirgyp, r trvjri tzgyvi, rcjf befne rj trvjrij tzgyvi, kyv jyzwk tzgyvi,
    trvjrij ...
```

```
Caesar 02: yd shofjewhqfxo, q squiqh syfxuh, qbie ademd qi squiqhi syfxuh, jxu ixyvj syfxuh,
    squiqhi ...
Caesar 01: xc rgneidvgpewn, p rpthpg rxewtg, pahd zcdlc ph rpthpgh rxewtg, iwt hwxui rxewtg,
    rpthpgh ...
Caesar 00: wb qfmdhcufodvm, o qosgof qwdvsf, ozgc ybckb og qosgofg qwdvsf, hvs gvwth qwdvsf,
    qosgofg ...
```

1.3 BASIC

1.3.1 Code

```
''Colin MacDonald
''Run this online: https://www.jdoodle.com/ia/b3V
Function encrypt(st As String, shiftAmount as Integer) As String
    Dim asciCode as Integer
    Dim result as String

    ''For char in str
    for i as Integer = 0 to len(st)-1
        asciCode = st[i]

        ''Shift letter (logic seperated by case)
        if(asciCode>=65) and (asciCode<=91) then
         result &= chr((asciCode + shiftAmount - 65) mod 26 + 65)
      elseif(asciCode>=97) and (asciCode<=122) then
         result &= chr((asciCode + shiftAmount - 97) mod 26 + 97)
      else
         result &= chr(asciCode)
      End if
    next i

    Return result
End Function

Function decrypt (st as String, shiftAmount as Integer) As String
    ''Cipher is cyclic, so run encrypt basically in reverse
   Return encrypt(st, 26-shiftAmount)
End Function

Sub solve (st as String, maxShiftValue as Integer)
    ''Call encrypt as much as requested
    for i as Integer = maxShiftValue to 0 step -1
        print "Caesar " & Str(i) & ": " & encrypt(st, i)
    next i
End Sub

''Input to test
Dim x as String = encrypt("This is a test string from Alan", 8)
print x
x = decrypt(x, 8)
print x
x =  encrypt("As with all single-alphabet substitution ciphers, the Caesar cipher is easily broken
     and in modern practice offers essentially no communications security.", 19)
print x
```

```
x = decrypt(x, 19)
print x

solve("HAL", 26)
solve("wb qfmdhcufodvm, o qosgof qwdvsf, ozgc ybckb og qosgofg qwdvsf, hvs gvwth qwdvsf, qosgofg
    qcrs cf qosgof gvwth, wg cbs ct hvs gwadzsgh obr acgh kwrszm ybckb sbqfmdhwcb hsqvbweisg.",14)
```

1.3.2 Output

```
Bpqa qa i bmab abzqvo nzwu Itiv
This is a test string from Alan
Tl pbma tee lbgzex-teiatuxm lnulmbmnmbhg vbiaxkl, max Vtxltk vbiaxk bl xtlber ukhdxg tgw bg fhwxkg
    iktvmbvx hyyxkl xllxgmbteer gh vhffngbvtmbhgl lxvnkbmr.
As with all single-alphabet substitution ciphers, the Caesar cipher is easily broken and in modern
    practice offers essentially no communications security.
Caesar 26: HAL
Caesar 25: GZK
Caesar 24: FYJ
Caesar 23: EXI
Caesar 22: DWH
Caesar 21: CVG
Caesar 20: BUF
Caesar 19: ATE
Caesar 18: ZSD
Caesar 17: YRC
Caesar 16: XQB
Caesar 15: WPA
Caesar 14: VOZ
Caesar 13: UNY
Caesar 12: TMX
Caesar 11: SLW
Caesar 10: RKV
Caesar 9: QJU
Caesar 8: PIT
Caesar 7: OHS
Caesar 6: NGR
Caesar 5: MFQ
Caesar 4: LEP
Caesar 3: KDO
Caesar 2: JCN
Caesar 1: IBM
Caesar 0: HAL
Caesar 14: kp etarvqitcrja, c ecguct ekrjgt, cnuq mpqyp cu ecguctu ekrjgt, vjg ujkhv ekrjgt,
    ecguctu eqfg qt ecguct ujkhv, ku qpg qh vjg ukornguv cpf oquv ykfgna mpqyp gpetarvkqp
    vgejpkswgu.
Caesar 13: jo dszquphsbqiz, b dbftbs djqifs, bmtp lopxo bt dbftbst djqifs, uif tijgu djqifs,
    dbftbst dpef ps dbftbs tijgu, jt pof pg uif tjnqmftu boe nptu xjefmz lopxo fodszqujpo
    ufdiojrvft.
Caesar 12: in cryptography, a caesar cipher, also known as caesars cipher, the shift cipher,
    caesars code or caesar shift, is one of the simplest and most widely known encryption
    techniques.
Caesar 11: hm bqxosnfqzogx, z bzdrzq bhogdq, zkrn jmnvm zr bzdrzqr bhogdq, sgd rghes bhogdq,
    bzdrzqr bncd nq bzdrzq rghes, hr nmd ne sgd rhlokdrs zmc lnrs vhcdkx jmnvm dmbqxoshnm
    sdbgmhptdr.
```

```
Caesar 10: gl apwnrmepynfw, y aycqyp agnfcp, yjqm ilmul yq aycqypq agnfcp, rfc qfgdr agnfcp,
    aycqypq ambc mp aycqyp qfgdr, gq mlc md rfc qgknjcqr ylb kmqr ugbcjw ilmul clapwnrgml
    rcaflgoscq.
Caesar 9: fk zovmqldoxmev, x zxbpxo zfmebo, xipl hkltk xp zxbpxop zfmebo, qeb pefcq zfmebo, zxbpxop
    zlab lo zxbpxo pefcq, fp lkb lc qeb pfjmibpq xka jlpq tfabiv hkltk bkzovmqflk qbzekfnrbp.
Caesar 8: ej ynulpkcnwldu, w ywaown yeldan, whok gjksj wo ywaowno yeldan, pda odebp yeldan, ywaowno
    ykza kn ywaown odebp, eo kja kb pda oeilhaop wjz ikop sezahu gjksj ajynulpekj paydjemqao.
Caesar 7: di xmtkojbmvkct, v xvznvm xdkczm, vgnj fijri vn xvznvmn xdkczm, ocz ncdao xdkczm, xvznvmn
    xjyz jm xvznvm ncdao, dn jiz ja ocz ndhkgzno viy hjno rdyzgt fijri zixmtkodji ozxcidlpzn.
Caesar 6: ch wlsjnialujbs, u wuymul wcjbyl, ufmi ehiqh um wuymulm wcjbyl, nby mbczn wcjbyl, wuymulm
    wixy il wuymul mbczn, cm ihy iz nby mcgjfymn uhx gimn qcxyfs ehiqh yhwlsjncih nywbhckoym.
Caesar 5: bg vkrimhzktiar, t vtxltk vbiaxk, telh dghpg tl vtxltkl vbiaxk, max labym vbiaxk, vtxltkl
    vhwx hk vtxltk labym, bl hgx hy max lbfiexlm tgw fhlm pbwxer dghpg xgvkrimbhg mxvagbjnxl.
Caesar 4: af ujqhlgyjshzq, s uswksj uahzwj, sdkg cfgof sk uswksjk uahzwj, lzw kzaxl uahzwj, uswksjk
    ugvw gj uswksj kzaxl, ak gfw gx lzw kaehdwkl sfv egkl oavwdq cfgof wfujqhlagf lwuzfaimwk.
Caesar 3: ze tipgkfxirgyp, r trvjri tzgyvi, rcjf befne rj trvjrij tzgyvi, kyv jyzwk tzgyvi, trvjrij
    tfuv fi trvjri jyzwk, zj fev fw kyv jzdgcvjk reu dfjk nzuvcp befne vetipgkzfe kvtyezhlvj.
Caesar 2: yd shofjewhqfxo, q squiqh syfxuh, qbie ademd qi squiqhi syfxuh, jxu ixyvj syfxuh, squiqhi
    setu eh squiqh ixyvj, yi edu ev jxu iycfbuij qdt ceij mytubo ademd udshofjyed jusxdygkui.
Caesar 1: xc rgneidvgpewn, p rpthpg rxewtg, pahd zcdlc ph rpthpgh rxewtg, iwt hwxui rxewtg, rpthpgh
    rdst dg rpthpg hwxui, xh dct du iwt hxbeathi pcs bdhi lxstan zcdlc tcrgneixdc itrwcxfjth.
Caesar 0: wb qfmdhcufodvm, o qosgof qwdvsf, ozgc ybckb og qosgofg qwdvsf, hvs gvwth qwdvsf, qosgofg
    qcrs cf qosgof gvwth, wg cbs ct hvs gwadzsgh obr acgh kwrszm ybckb sbqfmdhwcb hsqvbweisg.
```

1.4 Pascal

1.4.1 Code

```
{Colin MacDonald}
{Run this online: https://ideone.com/dfmbg6}
program ideone;
Uses sysutils;

function encrypt (str:String; shiftAmount:integer): string;
var
result: string;
i: integer;
character: char;
begin
   result:='';
   {For each char in the string}
   for i:= 1 to length(str) do
   begin
      character := str[i];
      {Uppercase}
      if(ord(character)>=65) and (ord(character)<=91) then
         result := result + chr((ord(character) + shiftAmount-65) mod 26 + 65)
      {Lowercase}
      else if(ord(character)>=97) and (ord(character)<=122) then
         result := result + chr((ord(character) + shiftAmount - 97) mod 26 + 97)
      else
         result := result + character;
   end;
   encrypt:=result;
```

```
end;

function decrypt (str:String; shiftAmount:integer): string;
begin
   decrypt:=encrypt(str, 26-shiftAmount);
end;

procedure solve (str:String; maxShiftValue:integer);
var
   i: integer;
begin
   for i:=maxShiftValue downto 0 do
   begin
      writeln('Caesar ' + IntToStr(i) + ': ' + encrypt(str, i));
   end;
end;

var
x: string;
begin
   x := encrypt('This is a test string from Alan', 8);
   writeln(x);
   x := decrypt(x, 8);
   writeln(x);
   x :=  encrypt(
      'As with all single-alphabet substitution ciphers, the Caesar cipher is easily broken and in
          modern practice offers essentially no communications security.', 19
    );
    writeln(x);
    x := decrypt(x, 19);
   writeln(x);

   solve('HAL', 26);
   solve('wb qfmdhcufodvm, o qosgof qwdvsf, ozgc ybckb og qosgofg qwdvsf, hvs gvwth qwdvsf, qosgofg
       qcrs cf qosgof gvwth, wg cbs ct hvs gwadzsgh obr acgh kwrszm ybckb sbqfmdhwcb
       hsqvbweisg.',26)
end.
```

1.4.2 Output

```
Bpqa qa i bmab abzqvo nzwu Itiv
This is a test string from Alan
Tl pbma tee lbgzex-teiatuxm lnulmbmnmbhg vbiaxkl, max Vtxltk vbiaxk bl xtlber ukhdxg tgw bg fhwxkg
    iktvmbvx hyyxkl xllxgmbteer gh vhffngbvtmbhgl lxvnkbmr.
As with all single-alphabet substitution ciphers, the Caesar cipher is easily broken and in modern
    practice offers essentially no communications security.
Caesar 26: HAL
Caesar 25: GZK
Caesar 24: FYJ
Caesar 23: EXI
Caesar 22: DWH
Caesar 21: CVG
Caesar 20: BUF
Caesar 19: ATE
```

```
Caesar 18: ZSD
Caesar 17: YRC
Caesar 16: XQB
Caesar 15: WPA
Caesar 14: VOZ
Caesar 13: UNY
Caesar 12: TMX
Caesar 11: SLW
Caesar 10: RKV
Caesar 9: QJU
Caesar 8: PIT
Caesar 7: OHS
Caesar 6: NGR
Caesar 5: MFQ
Caesar 4: LEP
Caesar 3: KDO
Caesar 2: JCN
Caesar 1: IBM
Caesar 0: HAL
Caesar 26: wb qfmdhcufodvm, o qosgof qwdvsf, ozgc ybckb og qosgofg qwdvsf, hvs gvwth qwdvsf,
    qosgofg qcrs cf qosgof gvwth, wg cbs ct hvs gwadzsgh obr acgh kwrszm ybckb sbqfmdhwcb
    hsqvbweisg.
Caesar 25: va pelcgbtencul, n pnrfne pvcure, nyfb xabja nf pnrfnef pvcure, gur fuvsg pvcure,
    pnrfnef pbqr be pnrfne fuvsg, vf bar bs gur fvzcyrfg naq zbfg jvqryl xabja rapelcgvba
    grpuavdhrf.
Caesar 24: uz odkbfasdmbtk, m omqemd oubtqd, mxea wzaiz me omqemde oubtqd, ftq eturf oubtqd,
    omqemde oapq ad omqemd eturf, ue azq ar ftq euybxqef mzp yaef iupqxk wzaiz qzodkbfuaz
    fqotzucgqe.
Caesar 23: ty ncjaezrclasj, l nlpdlc ntaspc, lwdz vyzhy ld nlpdlcd ntaspc, esp dstqe ntaspc,
    nlpdlcd nzop zc nlpdlc dstqe, td zyp zq esp dtxawpde lyo xzde htopwj vyzhy pyncjaetzy
    epnsytbfpd.
Caesar 22: sx mbizdyqbkzri, k mkockb mszrob, kvcy uxygx kc mkockbc mszrob, dro crspd mszrob,
    mkockbc myno yb mkockb crspd, sc yxo yp dro cswzvocd kxn wycd gsnovi uxygx oxmbizdsyx
    domrxsaeoc.
Caesar 21: rw lahycxpajyqh, j ljnbja lryqna, jubx twxfw jb ljnbjab lryqna, cqn bqroc lryqna,
    ljnbjab lxmn xa ljnbja bqroc, rb xwn xo cqn brvyunbc jwm vxbc frmnuh twxfw nwlahycrxw
    cnlqwrzdnb.
Caesar 20: qv kzgxbwozixpg, i kimaiz kqxpmz, itaw svwev ia kimaiza kqxpmz, bpm apqnb kqxpmz,
    kimaiza kwlm wz kimaiz apqnb, qa wvm wn bpm aquxtmab ivl uwab eqlmtg svwev mvkzgxbqwv
    bmkpvqycma.
Caesar 19: pu jyfwavnyhwof, h jhlzhy jpwoly, hszv ruvdu hz jhlzhyz jpwoly, aol zopma jpwoly,
    jhlzhyz jvkl vy jhlzhy zopma, pz vul vm aol zptwslza huk tvza dpklsf ruvdu lujyfwapvu
    aljoupxblz.
Caesar 18: ot ixevzumxgvne, g igkygx iovnkx, gryu qtuct gy igkygxy iovnkx, znk ynolz iovnkx,
    igkygxy iujk ux igkygx ynolz, oy utk ul znk yosvrkyz gtj suyz cojkre qtuct ktixevzout
    zkintowaky.
Caesar 17: ns hwduytlwfumd, f hfjxfw hnumjw, fqxt pstbs fx hfjxfwx hnumjw, ymj xmnky hnumjw,
    hfjxfwx htij tw hfjxfw xmnky, nx tsj tk ymj xnruqjxy fsi rtxy bnijqd pstbs jshwduynts
    yjhmsnvzjx.
Caesar 16: mr gvctxskvetlc, e geiwev gmtliv, epws orsar ew geiwevw gmtliv, xli wlmjx gmtliv,
    geiwevw gshi sv geiwev wlmjx, mw sri sj xli wmqtpiwx erh qswx amhipc orsar irgvctxmsr
    xiglrmuyiw.
Caesar 15: lq fubswrjudskb, d fdhvdu flskhu, dovr nqrzq dv fdhvduv flskhu, wkh vkliw flskhu,
    fdhvduv frgh ru fdhvdu vkliw, lv rqh ri wkh vlpsohvw dqg prvw zlghob nqrzq hqfubswlrq
    whfkqltxhv.
```

```
Caesar 14: kp etarvqitcrja, c ecguct ekrjgt, cnuq mpqyp cu ecguctu ekrjgt, vjg ujkhv ekrjgt,
    ecguctu eqfg qt ecguct ujkhv, ku qpg qh vjg ukornguv cpf oquv ykfgna mpqyp gpetarvkqp
    vgejpkswgu.
Caesar 13: jo dszquphsbqiz, b dbftbs djqifs, bmtp lopxo bt dbftbst djqifs, uif tijgu djqifs,
    dbftbst dpef ps dbftbs tijgu, jt pof pg uif tjnqmftu boe nptu xjefmz lopxo fodszqujpo
    ufdiojrvft.
Caesar 12: in cryptography, a caesar cipher, also known as caesars cipher, the shift cipher,
    caesars code or caesar shift, is one of the simplest and most widely known encryption
    techniques.
Caesar 11: hm bqxosnfqzogx, z bzdrzq bhogdq, zkrn jmnvm zr bzdrzqr bhogdq, sgd rghes bhogdq,
    bzdrzqr bncd nq bzdrzq rghes, hr nmd ne sgd rhlokdrs zmc lnrs vhcdkx jmnvm dmbqxoshnm
    sdbgmhptdr.
Caesar 10: gl apwnrmepynfw, y aycqyp agnfcp, yjqm ilmul yq aycqypq agnfcp, rfc qfgdr agnfcp,
    aycqypq ambc mp aycqyp qfgdr, gq mlc md rfc qgknjcqr ylb kmqr ugbcjw ilmul clapwnrgml
    rcaflgoscq.
Caesar 9: fk zovmqldoxmev, x zxbpxo zfmebo, xipl hkltk xp zxbpxop zfmebo, qeb pefcq zfmebo, zxbpxop
    zlab lo zxbpxo pefcq, fp lkb lc qeb pfjmibpq xka jlpq tfabiv hkltk bkzovmqflk qbzekfnrbp.
Caesar 8: ej ynulpkcnwldu, w ywaown yeldan, whok gjksj wo ywaowno yeldan, pda odebp yeldan, ywaowno
    ykza kn ywaown odebp, eo kja kb pda oeilhaop wjz ikop sezahu gjksj ajynulpekj paydjemqao.
Caesar 7: di xmtkojbmvkct, v xvznvm xdkczm, vgnj fijri vn xvznvmn xdkczm, ocz ncdao xdkczm, xvznvmn
    xjyz jm xvznvm ncdao, dn jiz ja ocz ndhkgzno viy hjno rdyzgt fijri zixmtkodji ozxcidlpzn.
Caesar 6: ch wlsjnialujbs, u wuymul wcjbyl, ufmi ehiqh um wuymulm wcjbyl, nby mbczn wcjbyl, wuymulm
    wixy il wuymul mbczn, cm ihy iz nby mcgjfymn uhx gimn qcxyfs ehiqh yhwlsjncih nywbhckoym.
Caesar 5: bg vkrimhzktiar, t vtxltk vbiaxk, telh dghpg tl vtxltkl vbiaxk, max labym vbiaxk, vtxltkl
    vhwx hk vtxltk labym, bl hgx hy max lbfiexlm tgw fhlm pbwxer dghpg xgvkrimbhg mxvagbjnxl.
Caesar 4: af ujqhlgyjshzq, s uswksj uahzwj, sdkg cfgof sk uswksjk uahzwj, lzw kzaxl uahzwj, uswksjk
    ugvw gj uswksj kzaxl, ak gfw gx lzw kaehdwkl sfv egkl oavwdq cfgof wfujqhlagf lwuzfaimwk.
Caesar 3: ze tipgkfxirgyp, r trvjri tzgyvi, rcjf befne rj trvjrij tzgyvi, kyv jyzwk tzgyvi, trvjrij
    tfuv fi trvjri jyzwk, zj fev fw kyv jzdgcvjk reu dfjk nzuvcp befne vetipgkzfe kvtyezhlvj.
Caesar 2: yd shofjewhqfxo, q squiqh syfxuh, qbie ademd qi squiqhi syfxuh, jxu ixyvj syfxuh, squiqhi
    setu eh squiqh ixyvj, yi edu ev jxu iycfbuij qdt ceij mytubo ademd udshofjyed jusxdygkui.
Caesar 1: xc rgneidvgpewn, p rpthpg rxewtg, pahd zcdlc ph rpthpgh rxewtg, iwt hwxui rxewtg, rpthpgh
    rdst dg rpthpg hwxui, xh dct du iwt hxbeathi pcs bdhi lxstan zcdlc tcrgneixdc itrwcxfjth.
Caesar 0: wb qfmdhcufodvm, o qosgof qwdvsf, ozgc ybckb og qosgofg qwdvsf, hvs gvwth qwdvsf, qosgofg
    qcrs cf qosgof gvwth, wg cbs ct hvs gwadzsgh obr acgh kwrszm ybckb sbqfmdhwcb hsqvbweisg.
```

1.5 Scala

1.5.1 Code

```scala
//By Colin MacDonald
//Run this online: https://ideone.com/waaiOZ
object Main {
  val alph: String = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
  def main(args: Array[String]) {
    var x = encrypt("This is a test string from Alan", 8);
    println(x);
    var y = decrypt(x, 8);
    println(y);
    var xx = encrypt(
      "As with all single-alphabet substitution ciphers, the Caesar cipher is easily broken and in
          modern practice offers essentially no communications security.",
      19
    );
```

```
  println(xx);
  var yy = decrypt(xx, 19);
  println(yy);
  solve("HAL", 26);
  solve(
    "wb qfmdhcufodvm, o qosgof qwdvsf, ozgc ybckb og qosgof'g qwdvsf, hvs gvwth qwdvsf, qosgof'g
        qcrs cf qosgof gvwth, wg cbs ct hvs gwadzsgh obr acgh kwrszm ybckb sbqfmdhwcb
        hsqvbweisg.",
    26
  )
}

def incrementChar(char: Char): Char = {
  //If not in alphabet
  if (alph.indexOf(char) == -1) {
    return char;
  }
  //If at end of alphabet
  if (char == 'Z') {
    return 'A';
  }

  //Get current index and add 1
  val index = alph.indexOf(char);
  return alph(index + 1);
}

def decrementChar(char: Char): Char = {
  //If not in alphabet
  if (alph.indexOf(char) == -1) {
    return char;
  }
  //If at start of alphabet
  if (char == 'A') {
    return 'Z';
  }

  //Get current index and subtract 1
  val index = alph.indexOf(char);
  return alph(index - 1);
}

def encrypt(st: String, shiftAmount: Int): String = {
  var str = st.toUpperCase();
  var newStr = new Array[Char](str.length() + 1);
  //Convert string to array so that individual letters can be manipulated
  for (x <- 0 to str.length() - 1) {
    newStr(x) = str(x);
  }
  //For letter in str
  for (x <- 0 to str.length() - 1) {
    //Shift it
    for (y <- 1 to shiftAmount) {
      newStr(x) = incrementChar(newStr(x));
    }
  }
```

```
    //Convert array back to string so that it can be returned
    var newNewStr = "";
    for (x <- 0 to newStr.length() - 1) {
      newNewStr = newNewStr + newStr(x);
    }
    return newNewStr;
  }

  def decrypt(st: String, shiftAmount: Int): String = {
    var str = st.toUpperCase();
    var newStr = new Array[Char](str.length() + 1);
    //Convert string to array so that individual letters can be manipulated
    for (x <- 0 to str.length() - 1) {
      newStr(x) = str(x);
    }
    //For letter in str
    for (x <- 0 to str.length() - 1) {
      //Shift it
      for (y <- 1 to shiftAmount) {
        newStr(x) = decrementChar(newStr(x));
      }
    }
    var newNewStr = "";
    //Convert array back to string so that it can be returned
    for (x <- 0 to newStr.length() - 1) {
      newNewStr = newNewStr + newStr(x);
    }
    return newNewStr;
  }

  def solve(str: String, maxShiftValue: Int) = {
    //Run encrypt the the specified amount of times
    for (i <- maxShiftValue to 0 by -1) {
      println("Caesar " + i + ": " + encrypt(str, i));
    }
  }

}
```

1.5.2 Output

```
BPQA QA I BMAB ABZQVO NZWU ITIV
THIS IS A TEST STRING FROM ALAN
TL PBMA TEE LBGZEX-TEIATUXM LNULMBMNMBHG VBIAXKL, MAX VTXLTK VBIAXK BL XTLBER UKHDXG TGW BG FHWXKG
    IKTVMBVX HYYXKL XLLXGMBTEER GH VHFFNGBVTMBHGL LXVNKBMR.
AS WITH ALL SINGLE-ALPHABET SUBSTITUTION CIPHERS, THE CAESAR CIPHER IS EASILY BROKEN AND IN MODERN
    PRACTICE OFFERS ESSENTIALLY NO COMMUNICATIONS SECURITY.
Caesar 26: HAL
Caesar 25: GZK
Caesar 24: FYJ
Caesar 23: EXI
Caesar 22: DWH
Caesar 21: CVG
Caesar 20: BUF
```

Caesar 19: ATE
Caesar 18: ZSD
Caesar 17: YRC
Caesar 16: XQB
Caesar 15: WPA
Caesar 14: VOZ
Caesar 13: UNY
Caesar 12: TMX
Caesar 11: SLW
Caesar 10: RKV
Caesar 9: QJU
Caesar 8: PIT
Caesar 7: OHS
Caesar 6: NGR
Caesar 5: MFQ
Caesar 4: LEP
Caesar 3: KDO
Caesar 2: JCN
Caesar 1: IBM
Caesar 0: HAL
Caesar 26: WB QFMDHCUFODVM, O QOSGOF QWDVSF, OZGC YBCKB OG QOSGOF'G QWDVSF, HVS GVWTH QWDVSF, QOSGOF'G QCRS CF QOSGOF GVWTH, WG CBS CT HVS GWADZSGH OBR ACGH KWRSZM YBCKB SBQFMDHWCB HSQVBWEISG.
Caesar 25: VA PELCGBTENCUL, N PNRFNE PVCURE, NYFB XABJA NF PNRFNE'F PVCURE, GUR FUVSG PVCURE, PNRFNE'F PBQR BE PNRFNE FUVSG, VF BAR BS GUR FVZCYRFG NAQ ZBFG JVQRYL XABJA RAPELCGVBA GRPUAVDHRF.
Caesar 24: UZ ODKBFASDMBTK, M OMQEMD OUBTQD, MXEA WZAIZ ME OMQEMD'E OUBTQD, FTQ ETURF OUBTQD, OMQEMD'E OAPQ AD OMQEMD ETURF, UE AZQ AR FTQ EUYBXQEF MZP YAEF IUPQXK WZAIZ QZODKBFUAZ FQOTZUCGQE.
Caesar 23: TY NCJAEZRCLASJ, L NLPDLC NTASPC, LWDZ VYZHY LD NLPDLC'D NTASPC, ESP DSTQE NTASPC, NLPDLC'D NZOP ZC NLPDLC DSTQE, TD ZYP ZQ ESP DTXAWPDE LYO XZDE HTOPWJ VYZHY PYNCJAETZY EPNSYTBFPD.
Caesar 22: SX MBIZDYQBKZRI, K MKOCKB MSZROB, KVCY UXYGX KC MKOCKB'C MSZROB, DRO CRSPD MSZROB, MKOCKB'C MYNO YB MKOCKB CRSPD, SC YXO YP DRO CSWZVOCD KXN WYCD GSNOVI UXYGX OXMBIZDSYX DOMRXSAEOC.
Caesar 21: RW LAHYCXPAJYQH, J LJNBJA LRYQNA, JUBX TWXFW JB LJNBJA'B LRYQNA, CQN BQROC LRYQNA, LJNBJA'B LXMN XA LJNBJA BQROC, RB XWN XO CQN BRVYUNBC JWM VXBC FRMNUH TWXFW NWLAHYCRXW CNLQWRZDNB.
Caesar 20: QV KZGXBWOZIXPG, I KIMAIZ KQXPMZ, ITAW SVWEV IA KIMAIZ'A KQXPMZ, BPM APQNB KQXPMZ, KIMAIZ'A KWLM WZ KIMAIZ APQNB, QA WVM WN BPM AQUXTMAB IVL UWAB EQLMTG SVWEV MVKZGXBQWV BMKPVQYCMA.
Caesar 19: PU JYFWAVNYHWOF, H JHLZHY JPWOLY, HSZV RUVDU HZ JHLZHY'Z JPWOLY, AOL ZOPMA JPWOLY, JHLZHY'Z JVKL VY JHLZHY ZOPMA, PZ VUL VM AOL ZPTWSLZA HUK TVZA DPKLSF RUVDU LUJYFWAPVU ALJOUPXBLZ.
Caesar 18: OT IXEVZUMXGVNE, G IGKYGX IOVNKX, GRYU QTUCT GY IGKYGX'Y IOVNKX, ZNK YNOLZ IOVNKX, IGKYGX'Y IUJK UX IGKYGX YNOLZ, OY UTK UL ZNK YOSVRKYZ GTJ SUYZ COJKRE QTUCT KTIXEVZOUT ZKINTOWAKY.
Caesar 17: NS HWDUYTLWFUMD, F HFJXFW HNUMJW, FQXT PSTBS FX HFJXFW'X HNUMJW, YMJ XMNKY HNUMJW, HFJXFW'X HTIJ TW HFJXFW XMNKY, NX TSJ TK YMJ XNRUQJXY FSI RTXY BNIJQD PSTBS JSHWDUYNTS YJHMSNVZJX.
Caesar 16: MR GVCTXSKVETLC, E GEIWEV GMTLIV, EPWS ORSAR EW GEIWEV'W GMTLIV, XLI WLMJX GMTLIV, GEIWEV'W GSHI SV GEIWEV WLMJX, MW SRI SJ XLI WMQTPIWX ERH QSWX AMHIPC ORSAR IRGVCTXMSR XIGLRMUYIW.
Caesar 15: LQ FUBSWRJUDSKB, D FDHVDU FLSKHU, DOVR NQRZQ DV FDHVDU'V FLSKHU, WKH VKLIW FLSKHU, FDHVDU'V FRGH RU FDHVDU VKLIW, LV RQH RI WKH VLPSOHVW DQG PRVW ZLGHOB NQRZQ HQFUBSWLRQ WHFKQLTXHV.

Caesar 14: KP ETARVQITCRJA, C ECGUCT EKRJGT, CNUQ MPQYP CU ECGUCT'U EKRJGT, VJG UJKHV EKRJGT, ECGUCT'U EQFG QT ECGUCT UJKHV, KU QPG QH VJG UKORNGUV CPF OQUV YKFGNA MPQYP GPETARVKQP VGEJPKSWGU.

Caesar 13: JO DSZQUPHSBQIZ, B DBFTBS DJQIFS, BMTP LOPXO BT DBFTBS'T DJQIFS, UIF TIJGU DJQIFS, DBFTBS'T DPEF PS DBFTBS TIJGU, JT POF PG UIF TJNQMFTU BOE NPTU XJEFMZ LOPXO FODSZQUJPO UFDIOJRVFT.

Caesar 12: IN CRYPTOGRAPHY, A CAESAR CIPHER, ALSO KNOWN AS CAESAR'S CIPHER, THE SHIFT CIPHER, CAESAR'S CODE OR CAESAR SHIFT, IS ONE OF THE SIMPLEST AND MOST WIDELY KNOWN ENCRYPTION TECHNIQUES.

Caesar 11: HM BQXOSNFQZOGX, Z BZDRZQ BHOGDQ, ZKRN JMNVM ZR BZDRZQ'R BHOGDQ, SGD RGHES BHOGDQ, BZDRZQ'R BNCD NQ BZDRZQ RGHES, HR NMD NE SGD RHLOKDRS ZMC LNRS VHCDKX JMNVM DMBQXOSHNM SDBGMHPTDR.

Caesar 10: GL APWNRMEPYNFW, Y AYCQYP AGNFCP, YJQM ILMUL YQ AYCQYP'Q AGNFCP, RFC QFGDR AGNFCP, AYCQYP'Q AMBC MP AYCQYP QFGDR, GQ MLC MD RFC QGKNJCQR YLB KMQR UGBCJW ILMUL CLAPWNRGML RCAFLGOSCQ.

Caesar 9: FK ZOVMQLDOXMEV, X ZXBPXO ZFMEBO, XIPL HKLTK XP ZXBPXO'P ZFMEBO, QEB PEFCQ ZFMEBO, ZXBPXO'P ZLAB LO ZXBPXO PEFCQ, FP LKB LC QEB PFJMIBPQ XKA JLPQ TFABIV HKLTK BKZOVMQFLK QBZEKFNRBP.

Caesar 8: EJ YNULPKCNWLDU, W YWAOWN YELDAN, WHOK GJKSJ WO YWAOWN'O YELDAN, PDA ODEBP YELDAN, YWAOWN'O YKZA KN YWAOWN ODEBP, EO KJA KB PDA OEILHAOP WJZ IKOP SEZAHU GJKSJ AJYNULPEKJ PAYDJEMQAO.

Caesar 7: DI XMTKOJBMVKCT, V XVZNVM XDKCZM, VGNJ FIJRI VN XVZNVM'N XDKCZM, OCZ NCDAO XDKCZM, XVZNVM'N XJYZ JM XVZNVM NCDAO, DN JIZ JA OCZ NDHKGZNO VIY HJNO RDYZGT FIJRI ZIXMTKODJI OZXCIDLPZN.

Caesar 6: CH WLSJNIALUJBS, U WUYMUL WCJBYL, UFMI EHIQH UM WUYMUL'M WCJBYL, NBY MBCZN WCJBYL, WUYMUL'M WIXY IL WUYMUL MBCZN, CM IHY IZ NBY MCGJFYMN UHX GIMN QCXYFS EHIQH YHWLSJNCIH NYWBHCKOYM.

Caesar 5: BG VKRIMHZKTIAR, T VTXLTK VBIAXK, TELH DGHPG TL VTXLTK'L VBIAXK, MAX LABYM VBIAXK, VTXLTK'L VHWX HK VTXLTK LABYM, BL HGX HY MAX LBFIEXLM TGW FHLM PBWXER DGHPG XGVKRIMBHG MXVAGBJNXL.

Caesar 4: AF UJQHLGYJSHZQ, S USWKSJ UAHZWJ, SDKG CFGOF SK USWKSJ'K UAHZWJ, LZW KZAXL UAHZWJ, USWKSJ'K UGVW GJ USWKSJ KZAXL, AK GFW GX LZW KAEHDWKL SFV EGKL OAVWDQ CFGOF WFUJQHLAGF LWUZFAIMWK.

Caesar 3: ZE TIPGKFXIRGYP, R TRVJRI TZGYVI, RCJF BEFNE RJ TRVJRI'J TZGYVI, KYV JYZWK TZGYVI, TRVJRI'J TFUV FI TRVJRI JYZWK, ZJ FEV FW KYV JZDGCVJK REU DFJK NZUVCP BEFNE VETIPGKZFE KVTYEZHLVJ.

Caesar 2: YD SHOFJEWHQFXO, Q SQUIQH SYFXUH, QBIE ADEMD QI SQUIQH'I SYFXUH, JXU IXYVJ SYFXUH, SQUIQH'I SETU EH SQUIQH IXYVJ, YI EDU EV JXU IYCFBUIJ QDT CEIJ MYTUBO ADEMD UDSHOFJYED JUSXDYGKUI.

Caesar 1: XC RGNEIDVGPEWN, P RPTHPG RXEWTG, PAHD ZCDLC PH RPTHPG'H RXEWTG, IWT HWXUI RXEWTG, RPTHPG'H RDST DG RPTHPG HWXUI, XH DCT DU IWT HXBEATHI PCS BDHI LXSTAN ZCDLC TCRGNEIXDC ITRWCXFJTH.

Caesar 0: WB QFMDHCUFODVM, O QOSGOF QWDVSF, OZGC YBCKB OG QOSGOF'G QWDVSF, HVS GVWTH QWDVSF, QOSGOF'G QCRS CF QOSGOF GVWTH, WG CBS CT HVS GWADZSGH OBR ACGH KWRSZM YBCKB SBQFMDHWCB HSQVBWEISG.

---

2 Log and Commentary

2.0.1 Log

Date	Hours Spent	Tasks / Accomplishments / Issues / Thoughts
3-20-21	1hr 35min	Learned the basics of Scala and completed much of the Caesar Cipher with it.
3-21-20	2hr 30min	Completed the remaining part of Scala. Fully did Pascal.
3-22-20	1hr	Completed all of Basic.
3-25-20	2hr 12min	Started learning COBOL and began work on Encrypt.
3-26-20	5hr 10min	Completed COBOL. Did all of Fortran.

2.0.2 Commentary

3-20-21

Dear Diary, due to a combination of procrastination and lots of work in other classes, I am only just starting this project now. It is regretful that this is the case, but there is nothing to do but to complete the project and try to improve in the future.

I estimate that each programming language will take 3-5 hours to complete the Caesar Cipher, meaning that this project should take 15-25 hours to complete. I have decided to code the project in reverse order because the more modern languages should be more familiar.

I have started with Scala. The syntax reminds me a lot of TypeScript, with having to state the variable type after a colon. So far most of my searches have been about learning how the syntax for specific things works. 45 minutes in and I have a simple function which increments a character to the next character in the alphabet. 1hr30min and I have successfully made the encrypt function. It is somewhat messy because I decided to convert from a string to an array and then back again, but it works fine and that's what counts. I really don't like how for loops work. Another 5 minutes and I have the decrypt function working. As it is essentially the reverse of encrypt, I just copied all the encrypt code and changed a few things around. Maybe not an ideal programming practice but it gets the job done.

3-21-21

After spending about an hour, I have completed the solve method for Scala. I also used this time to format the code. In terms of readability and writability, it is comparable to modern programming languages. The biggest issue was figuring out the unique syntax, but once I got the basics down, I had no problem.

Now onto Pascal. This time I got smarter and looked up the general Caesar Cipher algorithm (I did not look up how to do it in Pascal). I realised that it is as simple as grabbing the asci value of each character in the string and shifting that accordingly. Then I did a lot of searching of how to use the Pascal syntax. I don't like the method of declaring code blocks with words, or needing to declare variables in a specific spot. Otherwise it isn't too bad. 50 minutes of work and I have a functioning encrypt method. Another few minutes and the decrypt method is operational. This time it works just by calling encrypt but having the shift amount be 26-shiftAmount; this is because the cipher is cyclic. Another 15 minutes and the solve method is complete. After a total of 1hr 30 min, Pascal is complete and logged in this document.

The writability of Pascal is similar to Scala, maybe a little worse. The need to have "begin" and "end" and "var" is annoying. However, that is nice in readability. It is easy to glance at a section and see the purpose of it. The writability might have been better if I was working in a proper IDE and had automatic word completion and indentation. Also the := took some getting used to.

3-22-21

Today I intend to complete BASIC. I have selected FreeBASIC, mostly because I can easily find an online compiler for it. After about 45 minutes I have the encrypt method working. I was able to adapt a lot of the code from my work with Pascal. Since this language is older, there is not as much information available from a simple search besides some old forum posts. Fortunately, the documentation on the FreeBASIC is comprehensive. 5 minutes later and some modifications to the Pascal code and the decrypt method is functional. Another 10 minutes and the solve method is complete. And with that, BASIC is complete.

So far it is my shortest program in terms of lines of code (also in time to complete, though this time I didn't really count my time writing this). I would say this is mostly due to my experience gained from working with Pascal and Scala, but partially due to Basic being good in terms of readability and writability. I think that it would be tough to write code for anything complicated with BASIC. But for a simple program like a Caesar Cipher, it was easy to write. BASIC also has good readability because the syntax is consistent and works well with indentations. Despite being an older language, I would rank it right alongside Scala and Pascal.

3-25-21

Now onto COBOL. Thanks to other work and career fair related activities, I was unable to get any work done on this the last 2 days, and am now cutting it very close. I have begun research of how to code in COBOL and already I hate it. It is nothing like any other programming language and there are just too many rules and things that can be done. I guess SQL would be the best comparison and you can do complicated things with simple statements. With COBOL, it seems you do simple things with complicated statements.

1hr 5min and I can convert from a character to ASCII and back again. Obviously, progress is very much slowed compared to before. 2hr 12min in and I almost have code that encrypts. There is no string concatenation and it is not in a function but the math works.

3-26-21

It's the final stretch and not looking too great for me. 20 minutes of work and I figured out concatenation. At 1hr 40min I have put the pieces from earlier together and have a working encrypt procedure that "returns" a value. At 2hr 20min I almost have a decrypt procedure. I am having issues passing parameters. At 2hr 40 min I'm able to get encrypt and decrypt working together. It turns out that user defined functions were the way to go instead of procedures. I went with procedures at first because there was much more info available about them. After 3hr 40min of work today, I finaly have a complete COBOL program that meets the requirements. I spent a total of 5hr 50 minutes on this, and hated every minute of it.

I think COBOL has poor readability and writability. The only good part is that individual lines are very readable because it is almost english. Everything else is bad. There are just too many rules to follow, too many options, and just overall confusing. It is not built like a modern language so it was hard to jump into. A lot of features I am used to are missing, such as simple loops, and variably length strings. Declaring variables was one of the most confusing aspects as well, and passing them as parameters was tricky. I will say, it was extremely satisfying when I figured out an issue after being stuck on it for a while. I hope to never code with COBOL again.

For Fortran, I didn't record my progress along the way as I am nearing the deadline and am in crunch time mode. It took me about 2hr 30min from start to finish. Honestly, Fortran wasn't too bad, especially compared to COBOL. It is somewhat similar to the first three languages I did with structure and syntax; COBOL is just way too different. Readability and writability is ok. It mostly behaves like a typical language. It has typical if statements and the do and do while loops are simple to use. The variable declaration is a little unique but it not bad to read or write. I don't really like the syntax for printing to console but it works. The only thing I couldn't figure out was how to remove the padding added when printing a number.

Final Thoughts

The whole project took about 12hr 30min, plus time spent figuring out LaTeX and putting this document together, which adds maybe another hour or two. Dividing by 5, each language took an average of 2.5 hours,

which beat my 3-5 hours estimate. Scala took 2.5hrs. Pascal took 1.5 hrs. Basic took 1hr. COBOL took almost 6hrs. Fortran took 2.5hrs.

Scala took a while because it was completely new and I had to start from the ground up. Pascal was an hour less because I looked up how to write a Caesar Cipher algorithm (without copping code), because it had similarities to Scala, and because I was in the mindset of learning a new language. BASIC was the quickest because of the experience I gained from the first 2 languages, and also because it is somewhat simple. COBOL was the longest because it is an awful language, was totally different from anything I have ever seen, and needed so much debugging. Fortran took an average amount of time because it is tricky in some areas, but didn't take forever because I had gained experience from the others.

This discrepancy can be explained because I purposely overestimated and I thought most of the languages would be as hard as COBOL or Fortran. I just took a wild guess and picked 3-5 hrs each because it sounded good. Originally I was thinking 3-4hrs but I thought I would give myself more wiggle room.

For ranking the languages, Scala, Pascal, and BASIC are at the top but not in any specific order. If I had to pick, maybe I would put BASIC in first. Fortran is after those three, in the middle. COBOL is in dead last.

I'm not sure the best way to approach including my internet searches, as they are numerous. Scala has 117 searches, Pascal has 103, BASIC has 58, COBOL has 186, Fortran has 80. Most of it was searching for something that I knew was possible, such as "for loop" or "string concatenation" or "get char location in string" and the name of the language, and some of it was how the specific syntax works such as "variable declaration" or "if statement" or "modulus". Also, I did a lot of ctrl+f searching in various documentation that I found.

Well, that's it. It was very interesting to see that most of the basic concepts translate between languages, even if they look different. My time tracking strategy is something I have never done before. I am typically bad at estimates so I might track my time on future projects so I can reflect on it. Overall, I liked this project, except for COBOL.