Programming in the Past

Robert Perrone

robert.perrone1@marist.edu

March 27, 2021

1 Fortran

1.1 Consulting Log

Expected hours needed: 10

Date	***Hours Spent***	***Tasks / Accomplishments / Issues / Thoughts***
03/02/2021	3	I began to get some of the basics down here and actually got rid of some errors that would not let me print. I think the hardest part is reading these ambiguous and long error messages.
03/08/2021	3	Made some progress with functions. It's still giving me trouble but I switched over to a different web based IDE that has some better error messaging because I found ideone.com to be confusing in error messaging with too much on the screen.
03/22/2021	1	I messed around with some things and got frustrated so I moved to start COBOL in order to not waste time dwelling on Fortran.
03/23/2021	2	I finally got the syntax down for what I was doing wrong in the function sections with some new found understanding for what was going on and what some keywords meant. Once the encrypt function was working, decrypt fell into place and then solve came shortly after. The most frustrating thing at the very end was why there were extraneous spaces in the middle of the "Caesar 25-0: ..." part in solve. I could not figure out how to reformat it, but it works and that's what matters.

1.2 Commentary

Fortran was very frustrating for me manly because of the syntax. I found it confusing and not a whole lot of useful documentation that worked in this version. The coding ground website was also more helpful but only for fortran because the error messages were easier to follow, but still confusing compared to something like C or Java. Overall it was not terrible, but not enjoyable.

1.3 Source Code

```
!Encrypt
function encrypt(testStr, shift) result(encryptedStr)
        implicit none

        !local variables
        character(len = 30), intent(in) :: testStr
        integer, intent(in) :: shift

        !character, dimension(30) :: encryptedChrArr
        character :: letter
        integer :: asciiVal, e

        character(len = 30) :: encryptedStr

        encryptedStr = trim(testStr)
        eloop: do e = 1, len(encryptedStr)
            !if character from A to Z
                letter = encryptedStr(e:e)
                asciiVal = ichar(letter)
                if((asciiVal >= 65) .and. (asciiVal <= 90)) then
                        if((asciiVal == 65) .and. (shift < 0)) then
                                encryptedStr(e:e) = char(asciiVal + 26 + shift)
                        else if((asciiVal == 90 ) .and. (shift > 0)) then
                                encryptedStr(e:e) = char(asciiVal - 26 + shift)
                        else
                                encryptedStr(e:e) = char(asciiVal + shift)
                        end if
                !if character from a to z
                else if((asciiVal >= 97) .and. (asciiVal <= 122)) then
                        if((asciiVal == 97) .and. (shift < 0)) then
                                encryptedStr(e:e) = char(asciiVal + 26 + shift)
                        else if((asciiVal == 122) .and. (shift > 0)) then
                                encryptedStr(e:e) = char(asciiVal - 26 + shift)
                        else
                                encryptedStr(e:e) = char(asciiVal + shift)
                        end if
                else
                    encryptedStr(e:e) = letter
                end if
        end do eloop
end function encrypt

!Decrypt
function decrypt(testStr, shift) result(decryptedStr)
        implicit none

        !local variables
        character(len = 30), intent(in) :: testStr
        integer, intent(in) :: shift

        character :: letter
        integer :: asciiVal, d

        character(len = 30) :: decryptedStr

        decryptedStr = trim(testStr)
        dloop: do d = 1, len(decryptedStr)
            !if character from A to Z
                letter = decryptedStr(d:d)
                asciiVal = ichar(letter)
                if((asciiVal >= 65) .and. (asciiVal <= 90)) then
                        if((asciiVal == 65) .and. (-shift < 0)) then
                                decryptedStr(d:d) = char(asciiVal + 26 - shift)
```

```
                        else if((asciiVal == 90 ) .and. (-shift > 0)) then
                                decryptedStr(d:d) = char(asciiVal - 26 - shift)
                        else
                                decryptedStr(d:d) = char(asciiVal - shift)
                        end if
                !if character from a to z
                else if((asciiVal >= 97) .and. (asciiVal <= 122)) then
                        if((asciiVal == 97) .and. (-shift < 0)) then
                                decryptedStr(d:d) = char(asciiVal + 26 - shift)
                        else if((asciiVal == 122) .and. (-shift > 0)) then
                                decryptedStr(d:d) = char(asciiVal - 26 - shift)
                        else
                                decryptedStr(d:d) = char(asciiVal - shift)
                        end if
                else
                    decryptedStr(d:d) = letter
                end if
        end do dloop
end function decrypt

!Solve
function solve(testStr, maxShift) result(solvedStr)
        implicit none

        !local variables
        character(len = 30), intent(in) :: testStr
        integer, intent(in) :: maxShift

        !character, dimension(30) :: solvedChrArr
        character :: letter
        integer :: shift, caesarNum
        integer :: asciiVal, j, s

        character(len = 30) :: solvedStr

        shift = -1
        solvedStr = trim(testStr)
        PRINT *, char(9), 'Caesar 26: ', solvedStr
        jloop: do j = 0, maxShift-1
        sloop: do s = 1, len(solvedStr)
            !if character from A to Z
                letter = solvedStr(s:s)
                asciiVal = ichar(letter)
                if((asciiVal >= 65) .and. (asciiVal <= 90)) then
                        if((asciiVal == 65) .and. (shift < 0)) then
                                solvedStr(s:s) = char(asciiVal + 26 + shift)
                        else if((asciiVal == 90 ) .and. (shift > 0)) then
                                solvedStr(s:s) = char(asciiVal - 26 + shift)
                        else
                                solvedStr(s:s) = char(asciiVal + shift)
                        end if
                !if character from a to z
                else if((asciiVal >= 97) .and. (asciiVal <= 122)) then
                        if((asciiVal == 97) .and. (shift < 0)) then
                                solvedStr(s:s) = char(asciiVal + 26 + shift)
                        else if((asciiVal == 122) .and. (shift > 0)) then
                                solvedStr(s:s) = char(asciiVal - 26 + shift)
                        else
                                solvedStr(s:s) = char(asciiVal + shift)
                        end if
                else
                    solvedStr(s:s) = letter
                end if
        end do sloop
        caesarNum = maxShift-j-1
```

```
        print *, char(9), 'Caesar ', caesarNum, ': ', solvedStr
    end do jloop
end function solve

program MAIN
!makes sure that variables "i,j,k,l,m,n" are not default to integer
        implicit none

!Declare variables
        character(len = 30) :: encrypt !return type
        character(len = 30) :: decrypt !return type
        character(len = 30) :: solve   !return type

        character(len = 30) :: testStr1, testStr2, testStr3, testStr4, testStr5, testStr6, testStr7, testSt
        character(len = 30), dimension(10) :: testStrArr(8) !one dimensional array
        character(len = 30) :: tempStr, encryptedTempStr, decryptedTempStr, solvedTempStr

        integer :: shiftAmount
        integer :: maxShiftAmount
        integer :: i

!Initialize variables
        testStr1 = 'IBM'
        testStr2 = 'Hello World'
        testStr3 = 'This is a test'
        testStr4 = 'Hi my name is Robbie'
        testStr5 = 'WandaVision'
        testStr6 = 'Abed'
        testStr7 = 'The Mandalorian'
        testStr8 = 'Wow I learned Fortran I think'

        shiftAmount = -1
        maxShiftAmount = 26


        testStrArr = (/ testStr1,testStr2,testStr3,testStr4,testStr5,testStr6,testStr7,testStr8 /)

        PRINT *, 'Caesar Cipher'
        PRINT *, ''

!Do loop to encrypt, decrypt, and solve
        iloop: do i = 1, SIZE(testStrArr)
                tempStr = (testStrArr(i))
                PRINT *, 'Original String: ', tempStr

                encryptedTempStr = encrypt(tempStr, shiftAmount)
                PRINT *, 'Encrypted String: ', encryptedTempStr
                decryptedTempStr = decrypt(encryptedTempStr, shiftAmount)
                PRINT *, 'Decrpyted String: ', decryptedTempStr

                PRINT *, 'Solve: '
                solvedTempStr = solve(encryptedTempStr, maxShiftAmount)
                PRINT *, ''
        end do iloop
end program MAIN
```

1.4 Output

```
 Caesar Cipher

 Original String: IBM
 Encrypted String: HAL
 Decrpyted String: IBM
```

```
Solve:
        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

Original String: Hello World
Encrypted String: Gdkkn Vnqkc
Decrpyted String: Hello World
Solve:
        Caesar 26: Gdkkn Vnqkc
        Caesar            25 : Fcjjm Umpjb
        Caesar            24 : Ebiil Tloia
        Caesar            23 : Dahhk Sknhz
        Caesar            22 : Czggj Rjmgy
        Caesar            21 : Byffi Qilfx
        Caesar            20 : Axeeh Phkew
        Caesar            19 : Zwddg Ogjdv
        Caesar            18 : Yvccf Nficu
        Caesar            17 : Xubbe Mehbt
        Caesar            16 : Wtaad Ldgas
        Caesar            15 : Vszzc Kcfzr
        Caesar            14 : Uryyb Jbeyq
        Caesar            13 : Tqxxa Iadxp
        Caesar            12 : Spwwz Hzcwo
        Caesar            11 : Rovvy Gybvn
        Caesar            10 : Qnuux Fxaum
        Caesar             9 : Pmttw Ewztl
        Caesar             8 : Olssv Dvysk
        Caesar             7 : Nkrru Cuxrj
        Caesar             6 : Mjqqt Btwqi
        Caesar             5 : Lipps Asvph
        Caesar             4 : Khoor Zruog
        Caesar             3 : Jgnnq Yqtnf
        Caesar             2 : Ifmmp Xpsme
        Caesar             1 : Hello World
        Caesar             0 : Gdkkn Vnqkc

Original String: This is a test
Encrypted String: Sghr hr z sdrs
Decrpyted String: This is a test
Solve:
```

```
        Caesar 26: Sghr hr z sdrs
        Caesar           25 : Rfgq gq y rcqr
        Caesar           24 : Qefp fp x qbpq
        Caesar           23 : Pdeo eo w paop
        Caesar           22 : Ocdn dn v ozno
        Caesar           21 : Nbcm cm u nymn
        Caesar           20 : Mabl bl t mxlm
        Caesar           19 : Lzak ak s lwkl
        Caesar           18 : Kyzj zj r kvjk
        Caesar           17 : Jxyi yi q juij
        Caesar           16 : Iwxh xh p ithi
        Caesar           15 : Hvwg wg o hsgh
        Caesar           14 : Guvf vf n grfg
        Caesar           13 : Ftue ue m fqef
        Caesar           12 : Estd td l epde
        Caesar           11 : Drsc sc k docd
        Caesar           10 : Cqrb rb j cnbc
        Caesar            9 : Bpqa qa i bmab
        Caesar            8 : Aopz pz h alza
        Caesar            7 : Znoy oy g zkyz
        Caesar            6 : Ymnx nx f yjxy
        Caesar            5 : Xlmw mw e xiwx
        Caesar            4 : Wklv lv d whvw
        Caesar            3 : Vjku ku c vguv
        Caesar            2 : Uijt jt b uftu
        Caesar            1 : This is a test
        Caesar            0 : Sghr hr z sdrs

Original String: Hi my name is Robbie
Encrypted String: Gh lx mzld hr Qnaahd
Decrpyted String: Hi my name is Robbie
Solve:
        Caesar 26: Gh lx mzld hr Qnaahd
        Caesar           25 : Fg kw lykc gq Pmzzgc
        Caesar           24 : Ef jv kxjb fp Olyyfb
        Caesar           23 : De iu jwia eo Nkxxea
        Caesar           22 : Cd ht ivhz dn Mjwwdz
        Caesar           21 : Bc gs hugy cm Livvcy
        Caesar           20 : Ab fr gtfx bl Khuubx
        Caesar           19 : Za eq fsew ak Jgttaw
        Caesar           18 : Yz dp erdv zj Ifsszv
        Caesar           17 : Xy co dqcu yi Herryu
        Caesar           16 : Wx bn cpbt xh Gdqqxt
        Caesar           15 : Vw am boas wg Fcppws
        Caesar           14 : Uv zl anzr vf Eboovr
        Caesar           13 : Tu yk zmyq ue Dannuq
        Caesar           12 : St xj ylxp td Czmmtp
        Caesar           11 : Rs wi xkwo sc Byllso
        Caesar           10 : Qr vh wjvn rb Axkkrn
        Caesar            9 : Pq ug vium qa Zwjjqm
        Caesar            8 : Op tf uhtl pz Yviipl
        Caesar            7 : No se tgsk oy Xuhhok
        Caesar            6 : Mn rd sfrj nx Wtggnj
        Caesar            5 : Lm qc reqi mw Vsffmi
        Caesar            4 : Kl pb qdph lv Ureelh
        Caesar            3 : Jk oa pcog ku Tqddkg
        Caesar            2 : Ij nz obnf jt Spccjf
        Caesar            1 : Hi my name is Robbie
        Caesar            0 : Gh lx mzld hr Qnaahd

Original String: WandaVision
Encrypted String: VzmczUhrhnm
Decrpyted String: WandaVision
Solve:
        Caesar 26: VzmczUhrhnm
```

```
        Caesar           25 : UylbyTgqgml
        Caesar           24 : TxkaxSfpflk
        Caesar           23 : SwjzwReoekj
        Caesar           22 : RviyvQdndji
        Caesar           21 : QuhxuPcmcih
        Caesar           20 : PtgwtOblbhg
        Caesar           19 : OsfvsNakagf
        Caesar           18 : NreurMzjzfe
        Caesar           17 : MqdtqLyiyed
        Caesar           16 : LpcspKxhxdc
        Caesar           15 : KobroJwgwcb
        Caesar           14 : JnaqnIvfvba
        Caesar           13 : ImzpmHueuaz
        Caesar           12 : HlyolGtdtzy
        Caesar           11 : GkxnkFscsyx
        Caesar           10 : FjwmjErbrxw
        Caesar            9 : EivliDqaqwv
        Caesar            8 : DhukhCpzpvu
        Caesar            7 : CgtjgBoyout
        Caesar            6 : BfsifAnxnts
        Caesar            5 : AerheZmwmsr
        Caesar            4 : ZdqgdYlvlrq
        Caesar            3 : YcpfcXkukqp
        Caesar            2 : XboebWjtjpo
        Caesar            1 : WandaVision
        Caesar            0 : VzmczUhrhnm

Original String: Abed
Encrypted String: Zadc
Decrpyted String: Abed
Solve:
        Caesar 26: Zadc
        Caesar           25 : Yzcb
        Caesar           24 : Xyba
        Caesar           23 : Wxaz
        Caesar           22 : Vwzy
        Caesar           21 : Uvyx
        Caesar           20 : Tuxw
        Caesar           19 : Stwv
        Caesar           18 : Rsvu
        Caesar           17 : Qrut
        Caesar           16 : Pqts
        Caesar           15 : Opsr
        Caesar           14 : Norq
        Caesar           13 : Mnqp
        Caesar           12 : Lmpo
        Caesar           11 : Klon
        Caesar           10 : Jknm
        Caesar            9 : Ijml
        Caesar            8 : Hilk
        Caesar            7 : Ghkj
        Caesar            6 : Fgji
        Caesar            5 : Efih
        Caesar            4 : Dehg
        Caesar            3 : Cdgf
        Caesar            2 : Bcfe
        Caesar            1 : Abed
        Caesar            0 : Zadc

Original String: The Mandalorian
Encrypted String: Sgd Lzmczknqhzm
Decrpyted String: The Mandalorian
Solve:
        Caesar 26: Sgd Lzmczknqhzm
        Caesar           25 : Rfc Kylbyjmpgyl
```

```
        Caesar           24 : Qeb Jxkaxilofxk
        Caesar           23 : Pda Iwjzwhknewj
        Caesar           22 : Ocz Hviyvgjmdvi
        Caesar           21 : Nby Guhxufilcuh
        Caesar           20 : Max Ftgwtehkbtg
        Caesar           19 : Lzw Esfvsdgjasf
        Caesar           18 : Kyv Dreurcfizre
        Caesar           17 : Jxu Cqdtqbehyqd
        Caesar           16 : Iwt Bpcspadgxpc
        Caesar           15 : Hvs Aobrozcfwob
        Caesar           14 : Gur Znaqnybevna
        Caesar           13 : Ftq Ymzpmxadumz
        Caesar           12 : Esp Xlyolwzctly
        Caesar           11 : Dro Wkxnkvybskx
        Caesar           10 : Cqn Vjwmjuxarjw
        Caesar            9 : Bpm Uivlitwzqiv
        Caesar            8 : Aol Thukhsvyphu
        Caesar            7 : Znk Sgtjgruxogt
        Caesar            6 : Ymj Rfsifqtwnfs
        Caesar            5 : Xli Qerhepsvmer
        Caesar            4 : Wkh Pdqgdoruldq
        Caesar            3 : Vjg Ocpfcnqtkcp
        Caesar            2 : Uif Nboebmpsjbo
        Caesar            1 : The Mandalorian
        Caesar            0 : Sgd Lzmczknqhzm

Original String: Wow I learned Fortran I think
Encrypted String: Vnv H kdzqmdc Enqsqzm H sghmj
Decrpyted String: Wow I learned Fortran I think
Solve:
        Caesar 26: Vnv H kdzqmdc Enqsqzm H sghmj
        Caesar           25 : Umu G jcyplcb Dmprpyl G rfgli
        Caesar           24 : Tlt F ibxokba Cloqoxk F qefkh
        Caesar           23 : Sks E hawnjaz Bknpnwj E pdejg
        Caesar           22 : Rjr D gzvmizy Ajmomvi D ocdif
        Caesar           21 : Qiq C fyulhyx Zilnluh C nbche
        Caesar           20 : Php B extkgxw Yhkmktg B mabgd
        Caesar           19 : Ogo A dwsjfwv Xgjljsf A lzafc
        Caesar           18 : Nfn Z cvrievu Wfikire Z kyzeb
        Caesar           17 : Mem Y buqhdut Vehjhqd Y jxyda
        Caesar           16 : Ldl X atpgcts Udgigpc X iwxcz
        Caesar           15 : Kck W zsofbsr Tcfhfob W hvwby
        Caesar           14 : Jbj V yrnearq Sbegena V guvax
        Caesar           13 : Iai U xqmdzqp Radfdmz U ftuzw
        Caesar           12 : Hzh T wplcypo Qzcecly T estyv
        Caesar           11 : Gyg S vokbxon Pybdbkx S drsxu
        Caesar           10 : Fxf R unjawnm Oxacajw R cqrwt
        Caesar            9 : Ewe Q tmizvml Nwzbziv Q bpqvs
        Caesar            8 : Dvd P slhyulk Mvyayhu P aopur
        Caesar            7 : Cuc O rkgxtkj Luxzxgt O znotq
        Caesar            6 : Btb N qjfwsji Ktwywfs N ymnsp
        Caesar            5 : Asa M pievrih Jsvxver M xlmro
        Caesar            4 : Zrz L ohduqhg Iruwudq L wklqn
        Caesar            3 : Yqy K ngctpgf Hqtvtcp K vjkpm
        Caesar            2 : Xpx J mfbsofe Gpsusbo J uijol
        Caesar            1 : Wow I learned Fortran I think
        Caesar            0 : Vnv H kdzqmdc Enqsqzm H sghmj
```

2 COBOL

2.1 Consulting Log

Expected hours needed: 12

Date	*Hours Spent*	*Tasks / Accomplishments / Issues / Thoughts*
03/22/2021	4	Spent a lot of time looking over the basic syntax and learning the program structures with the different areas of the overall program.
03/23/2021	2	I learned a good amount about the tables and how they work. They are still a bit weird and I mostly understand it, but it makes less sense to me in this context and more sense in the other examples I've seen like being able to hold costumer data.
03/25/2021	4	I made some good progress and went step by step more than I have been with any of the other languages, seeing what worked and what didn't. It made things easier and seemingly take less time. I made good progress on the encrypt subprogram and now it's just time to debug it and make it better. I am still struggling on how to trim the strings down with the trailing spaces. Nothing is seeming to work in this version that I found. I also finished the decrypt function, got out the bugs and the solve function is done, but buggy.
03/26/2021	1	I got a lot more done yesterday than I thought I would. I am still having trouble with the solve function because it is only printing out the first 2 letters for each shift on every single string. The letters are correct in the shift, but most of the updated string is lost in the ether and I don't know why.

2.2 Commentary

I found COBOL to actually be slightly more tolerable than Fortran in some ways. Although this felt nothing like any of the other programming languages, I think that actually helped in some ways, because I did not get as hung up on trying to write it like the other languages. At the same time, the syntax is even more confusing than Fortran's in a lot of ways and parts of it I did not understand. I think it would not have been that bad, but the IF statements and PERFORM keyword did not work the same in the version on ideone.com as the resources I saw. This version that they had there was more frustrating to work with in many ways that would not have happened if it was a different version of COBOL. There did not seem to be else if statements or else statements. Unless I was just writing them wrong, they did not work in any of the ways I had tried. Perform also did not work for me outside of paragraphs.

2.3 Source Code

```
$ SET SOURCEFORMAT "FREE"

IDENTIFICATION DIVISION.
PROGRAM-ID. IDEONE.
AUTHOR. Robert Perrone.
DATE-WRITTEN. March 22nd 2021
*>ENVIRONMENT DIVISION.

DATA DIVISION.
WORKING-STORAGE SECTION.
*> Original String Table
01 testStrTable.*> PIC X(30).
        02 testStr1 PIC X(30) VALUE "IBM".
        02 testStr2 PIC X(30) VALUE "Hello World".
```

```
        02 testStr3 PIC X(30) VALUE "This is a test".
        02 testStr4 PIC X(30) VALUE "Hi my name is Robbie".
        02 testStr5 PIC X(30) VALUE "WandaVision".
        02 testStr6 PIC X(30) VALUE "Abed".
        02 testStr7 PIC X(30) VALUE "The Mandalorian".
        02 testStr8 PIC X(30) VALUE "Wow I learned COBOL I think".
01 FILLER REDEFINES testStrTable.
        02 testStrTb OCCURS 8 INDEXED BY I.
                03 testStrTb OCCURS 1 TIMES.
                        04 testStr PIC X(30).

*> Encrypted String Table
01 eTestStrTable PIC X(30).
        02 eTestStr1 PIC X(30) VALUE "".
        02 eTestStr2 PIC X(30) VALUE "".
        02 eTestStr3 PIC X(30) VALUE "".
        02 eTestStr4 PIC X(30) VALUE "".
        02 eTestStr5 PIC X(30) VALUE "".
        02 eTestStr6 PIC X(30) VALUE "".
        02 eTestStr7 PIC X(30) VALUE "".
        02 eTestStr8 PIC X(30) VALUE "".
01 FILLER REDEFINES eTestStrTable.
        02 eTestStrTb OCCURS 8 TIMES INDEXED BY J.
                03 eTestStrTb OCCURS 1 TIMES.
                        04 eTestStr PIC X(30).

01 shiftAmount PIC S9(4) VALUE -1.
01 maxShiftAmount PIC 9(2) VALUE 26.

01 tempStr PIC X(30) VALUE "".
01 encryptedStr PIC X(30) VALUE "".
01 decryptedStr PIC X(30) VALUE "".
01 solvedStr PIC X(30) VALUE "".

*>01 I PIC 9(1) VALUE 1.
*>01 J PIC 9(1) VALUE 1.

PROCEDURE DIVISION.

*>      your code goes here
        DISPLAY "Caesar Cipher ".
        DISPLAY "".

        SET I TO 1.
        SET J TO 1.
        begin.
                PERFORM iForLoop VARYING I FROM 1 BY 1 UNTIL I>8.
                STOP RUN.

        iForLoop.
                MOVE Function UPPER-CASE(testStr(I)) TO tempStr.
                DISPLAY "Original String: " tempStr.

                CALL "ENCRYPT" USING tempStr, shiftAmount, encryptedStr.
                DISPLAY "Encrypted String: "  encryptedStr.

                CALL "DECRYPT" USING encryptedStr, shiftAmount, decryptedStr.
                DISPLAY "Decrypted String: " decryptedStr.

                DISPLAY "Solve:".
                CALL "SOLVE" USING encryptedStr, maxShiftAmount, solvedStr.
                DISPLAY "".
                *>MOVE Function solve(encryptedTempStr, maxShiftAmount) TO decryptedTempStr.
END PROGRAM IDEONE.

```

```
*>Encrypt Subprogram
IDENTIFICATION DIVISION.
PROGRAM-ID. ENCRYPT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 letter PIC X(1).
01 newLetter PIC X(1).
01 asciiVal PIC 9(2).
01 newAsciiVal PIC 9(2).

01 E PIC 9(2) VALUE 1.
01 len PIC 9(3) VALUE 0.

LINKAGE SECTION.
01 tempStr PIC X(30).
01 shiftAmount PIC S9(4).
01 encryptedStr PIC X(30).

PROCEDURE DIVISION USING tempStr, shiftAmount, encryptedStr.
        *>MOVE "returned" TO encryptedStr.
        *>DISPLAY "ENCRYPT".
        begin.
                SET shiftAmount TO -1.
                MOVE Function LENGTH(tempStr) TO len.
                *>DISPLAY len.
                PERFORM eForLoop VARYING E FROM 1 BY 1 UNTIL E > len.
                        EXIT PROGRAM.
        eForLoop.
                MOVE tempStr(E:E) TO letter.
                *>DISPLAY letter.
                MOVE Function ORD(letter) TO asciiVal.
                *>COMPUTE asciiVal = asciiVal - 1.
                *>DISPLAY asciiVal.

                *> If character from A to Z
                IF (asciiVal >= 66) AND (asciiVal <= 91) THEN
                        *>If character is A and shift is negative, wrap around
                        IF (asciiVal = 66) AND (shiftAmount < 0) THEN
                                COMPUTE newAsciiVal = asciiVal + 26 + shiftAmount.
                                MOVE Function CHAR(newAsciiVal) TO newLetter.
                                MOVE newLetter TO encryptedStr(E:E).

                        *>If character is Z and shift is positive, wrap around
                        IF ((asciiVal = 91) AND (shiftAmount > 0)) THEN
                                COMPUTE newAsciiVal = asciiVal - 26 + shiftAmount.
                                MOVE Function CHAR(newAsciiVal) TO newLetter.
                                MOVE newLetter TO encryptedStr(E:E).
                        *>ELSE
                        IF NOT ((asciiVal = 66) AND (shiftAmount < 0)) AND NOT ((asciiVal = 91) AND (shiftA
                                *>DISPLAY letter.
                                *>DISPLAY asciiVal.
                                COMPUTE newAsciiVal = asciiVal + shiftAmount.
                                *>DISPLAY shiftAmount.
                                *>DISPLAY newAsciiVal.
                                MOVE Function CHAR(newAsciiVal) TO newLetter.
                                MOVE newLetter TO encryptedStr(E:E).
                                *>DISPLAY encryptedStr(E:E).

                IF NOT ((asciiVal >= 66) AND (asciiVal <= 91)) THEN
                        *>If character is space or other
                        MOVE letter TO encryptedStr(E:E).

END PROGRAM ENCRYPT.

*>Decrypt Subprogram
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID. DECRYPT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 letter PIC X(1).
01 newLetter PIC X(1).
01 asciiVal PIC 9(2).
01 newAsciiVal PIC 9(2).

01 D PIC 9(2) VALUE 1.
01 len PIC 9(3) VALUE 0.

LINKAGE SECTION.
01 encryptedStr PIC X(30).
01 shiftAmount PIC S9(4).
01 decryptedStr PIC X(30).

PROCEDURE DIVISION USING encryptedStr, shiftAmount, decryptedStr.
        *>MOVE "returned" TO encryptedStr.
        *>DISPLAY "DECRYPT".
        begin.
                MOVE Function LENGTH(encryptedStr) TO len.
                *>DISPLAY len.
                MULTIPLY shiftAmount BY -1 GIVING shiftAmount.
                PERFORM dForLoop VARYING D FROM 1 BY 1 UNTIL D > len.
                        EXIT PROGRAM.
        dForLoop.
                MOVE encryptedStr(D:D) TO letter.
                *>DISPLAY letter.
                MOVE Function ORD(letter) TO asciiVal.
                *>COMPUTE asciiVal = asciiVal - 1.
                *>DISPLAY asciiVal.
                *> If character from A to Z

                *>Make shift amount opposite sign


                IF (asciiVal) >= 66 AND (asciiVal <= 91) THEN
                        *>If character is A and shift is negative, wrap around
                        IF (asciiVal = 66) AND (shiftAmount < 0) THEN
                                COMPUTE newAsciiVal = asciiVal + 26 + shiftAmount.
                                MOVE Function CHAR(newAsciiVal) TO newLetter.
                                MOVE newLetter TO decryptedStr(D:D).

                        *>If character is Z and shift is positive, wrap around
                        IF (asciiVal = 91) AND (shiftAmount > 0) THEN
                                COMPUTE newAsciiVal = asciiVal - 26 + shiftAmount.
                                MOVE Function CHAR(newAsciiVal) TO newLetter.
                                MOVE newLetter TO decryptedStr(D:D).
                        *>ELSE
                        IF NOT ((asciiVal = 66) AND (shiftAmount < 0)) AND NOT ((asciiVal = 91) AND (shiftA
                                *>DISPLAY letter.
                                *>DISPLAY asciiVal.
                                COMPUTE newAsciiVal = asciiVal + shiftAmount.
                                *>DISPLAY shiftAmount.
                                *>DISPLAY newAsciiVal.
                                MOVE Function CHAR(newAsciiVal) TO newLetter.
                                MOVE newLetter TO decryptedStr(D:D).
                                *>DISPLAY encryptedStr(D:D).

                IF NOT ((asciiVal >= 66) AND (asciiVal <= 91)) THEN
                        *>If character is space or other
                        MOVE letter TO decryptedStr(D:D).

END PROGRAM DECRYPT.
```

```
210
211 *>Solve Subprogram
212 IDENTIFICATION DIVISION.
213 PROGRAM-ID. SOLVE.
214 DATA DIVISION.
215 WORKING-STORAGE SECTION.
216 01 letter PIC X(1).
217 01 newLetter PIC X(1).
218 01 asciiVal PIC 9(2).
219 01 newAsciiVal PIC 9(2).
220 01 caesarNum PIC 9(2).
221 01 shiftAmount PIC S9(4) VALUE -1.
222
223 01 J PIC 9(2) VALUE 1.
224 01 S PIC 9(2) VALUE 1.
225 01 len PIC 9(3) VALUE 0.
226
227 LINKAGE SECTION.
228 01 encryptedStr PIC X(30).
229 01 maxShiftAmount PIC 9(2).
230 01 solvedStr PIC X(30).
231
232 PROCEDURE DIVISION USING encryptedStr, maxShiftAmount, solvedStr.
233         *>MOVE "returned" TO solvedStr.
234         *>DISPLAY "SOLVE".
235         begin.
236                 MOVE Function LENGTH(solvedStr) TO len.
237                 SET shiftAmount TO -1.
238                 MOVE encryptedStr TO solvedStr.
239                 DISPLAY "Caesar 26: " solvedStr.
240                 PERFORM jForLoop VARYING J FROM 1 BY 1 UNTIL J > maxShiftAmount.
241                         EXIT PROGRAM.
242
243         jForLoop.
244                 PERFORM sForLoop VARYING S FROM 1 BY 1 UNTIL S > len.
245                 COMPUTE caesarNum = maxShiftAmount - j.
246                 DISPLAY "Caesar " caesarNum ": " solvedStr.
247
248         sForLoop.
249                 MOVE solvedStr(S:S) TO letter.
250                 *>DISPLAY letter.
251                 MOVE Function ORD(letter) TO asciiVal.
252                 *>COMPUTE asciiVal = asciiVal - 1.
253                 *>DISPLAY asciiVal.
254                 *> If character from A to Z
255                 IF (asciiVal >= 66) AND (asciiVal <= 91) THEN
256                         *>If character is A and shift is negative, wrap around
257                         IF (asciiVal = 66) AND (shiftAmount < 0) THEN
258                                 COMPUTE newAsciiVal = asciiVal + 26 + shiftAmount.
259                                 MOVE Function CHAR(newAsciiVal) TO newLetter.
260                                 MOVE newLetter TO solvedStr(S:S).
261                         *>If character is Z and shift is positive, wrap around
262                         IF (asciiVal = 91) AND (shiftAmount > 0) THEN
263                                 COMPUTE newAsciiVal = asciiVal - 26 + shiftAmount.
264                                 MOVE Function CHAR(newAsciiVal) TO newLetter.
265                                 MOVE newLetter TO solvedStr(S:S).
266                         *>ELSE
267                         IF NOT ((asciiVal = 66) AND (shiftAmount < 0)) AND NOT ((asciiVal = 91) AND (shiftA
268                                 *>DISPLAY letter.
269                                 *>DISPLAY asciiVal.
270                                 COMPUTE newAsciiVal = asciiVal + shiftAmount.
271                                 *>DISPLAY shiftAmount.
272                                 *>DISPLAY newAsciiVal.
273                                 MOVE Function CHAR(newAsciiVal) TO newLetter.
274                                 MOVE newLetter TO solvedStr(S:S).
```

```
                IF NOT ((asciiVal >= 66) AND (asciiVal <= 91)) THEN
                        *>If character is space or other
                        MOVE letter TO solvedStr(S:S).

END PROGRAM SOLVE.
```

2.4 Output

```
Caesar Cipher

Original String: IBM
Encrypted String: HAL
Decrypted String: IBM
Solve:
Caesar 26: HAL
Caesar 25: GZ
Caesar 24: FY
Caesar 23: EX
Caesar 22: DW
Caesar 21: CV
Caesar 20: BU
Caesar 19: AT
Caesar 18: ZS
Caesar 17: YR
Caesar 16: XQ
Caesar 15: WP
Caesar 14: VO
Caesar 13: UN
Caesar 12: TM
Caesar 11: SL
Caesar 10: RK
Caesar 09: QJ
Caesar 08: PI
Caesar 07: OH
Caesar 06: NG
Caesar 05: MF
Caesar 04: LE
Caesar 03: KD
Caesar 02: JC
Caesar 01: IB
Caesar 00: HA

Original String: HELLO WORLD
Encrypted String: GDKKN VNQKC
Decrypted String: HELLO WORLD
Solve:
Caesar 26: GDKKN VNQKC
Caesar 25: FC
Caesar 24: EB
Caesar 23: DA
Caesar 22: CZ
Caesar 21: BY
Caesar 20: AX
Caesar 19: ZW
Caesar 18: YV
Caesar 17: XU
Caesar 16: WT
Caesar 15: VS
Caesar 14: UR
Caesar 13: TQ
Caesar 12: SP
Caesar 11: RO
Caesar 10: QN
```

```
Caesar 09: PM
Caesar 08: OL
Caesar 07: NK
Caesar 06: MJ
Caesar 05: LI
Caesar 04: KH
Caesar 03: JG
Caesar 02: IF
Caesar 01: HE
Caesar 00: GD

Original String: THIS IS A TEST
Encrypted String: SGHR HR Z SDRS
Decrypted String: THIS IS A TEST
Solve:
Caesar 26: SGHR HR Z SDRS
Caesar 25: RF
Caesar 24: QE
Caesar 23: PD
Caesar 22: OC
Caesar 21: NB
Caesar 20: MA
Caesar 19: LZ
Caesar 18: KY
Caesar 17: JX
Caesar 16: IW
Caesar 15: HV
Caesar 14: GU
Caesar 13: FT
Caesar 12: ES
Caesar 11: DR
Caesar 10: CQ
Caesar 09: BP
Caesar 08: AO
Caesar 07: ZN
Caesar 06: YM
Caesar 05: XL
Caesar 04: WK
Caesar 03: VJ
Caesar 02: UI
Caesar 01: TH
Caesar 00: SG

Original String: HI MY NAME IS ROBBIE
Encrypted String: GH LX MZLD HR QNAAHD
Decrypted String: HI MY NAME IS ROBBIE
Solve:
Caesar 26: GH LX MZLD HR QNAAHD
Caesar 25: FG
Caesar 24: EF
Caesar 23: DE
Caesar 22: CD
Caesar 21: BC
Caesar 20: AB
Caesar 19: ZA
Caesar 18: YZ
Caesar 17: XY
Caesar 16: WX
Caesar 15: VW
Caesar 14: UV
Caesar 13: TU
Caesar 12: ST
Caesar 11: RS
Caesar 10: QR
Caesar 09: PQ
```

```
Caesar 08: OP
Caesar 07: NO
Caesar 06: MN
Caesar 05: LM
Caesar 04: KL
Caesar 03: JK
Caesar 02: IJ
Caesar 01: HI
Caesar 00: GH

Original String: WANDAVISION
Encrypted String: VZMCZUHRHNM
Decrypted String: WANDAVISION
Solve:
Caesar 26: VZMCZUHRHNM
Caesar 25: UY
Caesar 24: TX
Caesar 23: SW
Caesar 22: RV
Caesar 21: QU
Caesar 20: PT
Caesar 19: OS
Caesar 18: NR
Caesar 17: MQ
Caesar 16: LP
Caesar 15: KO
Caesar 14: JN
Caesar 13: IM
Caesar 12: HL
Caesar 11: GK
Caesar 10: FJ
Caesar 09: EI
Caesar 08: DH
Caesar 07: CG
Caesar 06: BF
Caesar 05: AE
Caesar 04: ZD
Caesar 03: YC
Caesar 02: XB
Caesar 01: WA
Caesar 00: VZ

Original String: ABED
Encrypted String: ZADC
Decrypted String: ABED
Solve:
Caesar 26: ZADC
Caesar 25: YZ
Caesar 24: XY
Caesar 23: WX
Caesar 22: VW
Caesar 21: UV
Caesar 20: TU
Caesar 19: ST
Caesar 18: RS
Caesar 17: QR
Caesar 16: PQ
Caesar 15: OP
Caesar 14: NO
Caesar 13: MN
Caesar 12: LM
Caesar 11: KL
Caesar 10: JK
Caesar 09: IJ
Caesar 08: HI
```

```
Caesar 07: GH
Caesar 06: FG
Caesar 05: EF
Caesar 04: DE
Caesar 03: CD
Caesar 02: BC
Caesar 01: AB
Caesar 00: ZA

Original String: THE MANDALORIAN
Encrypted String: SGD LZMCZKNQHZM
Decrypted String: THE MANDALORIAN
Solve:
Caesar 26: SGD LZMCZKNQHZM
Caesar 25: RF
Caesar 24: QE
Caesar 23: PD
Caesar 22: OC
Caesar 21: NB
Caesar 20: MA
Caesar 19: LZ
Caesar 18: KY
Caesar 17: JX
Caesar 16: IW
Caesar 15: HV
Caesar 14: GU
Caesar 13: FT
Caesar 12: ES
Caesar 11: DR
Caesar 10: CQ
Caesar 09: BP
Caesar 08: AO
Caesar 07: ZN
Caesar 06: YM
Caesar 05: XL
Caesar 04: WK
Caesar 03: VJ
Caesar 02: UI
Caesar 01: TH
Caesar 00: SG

Original String: WOW I LEARNED COBOL I THINK
Encrypted String: VNV H KDZQMDC BNANK H SGHMJ
Decrypted String: WOW I LEARNED COBOL I THINK
Solve:
Caesar 26: VNV H KDZQMDC BNANK H SGHMJ
Caesar 25: UM
Caesar 24: TL
Caesar 23: SK
Caesar 22: RJ
Caesar 21: QI
Caesar 20: PH
Caesar 19: OG
Caesar 18: NF
Caesar 17: ME
Caesar 16: LD
Caesar 15: KC
Caesar 14: JB
Caesar 13: IA
Caesar 12: HZ
Caesar 11: GY
Caesar 10: FX
Caesar 09: EW
Caesar 08: DV
Caesar 07: CU
```

```
Caesar 06: BT
Caesar 05: AS
Caesar 04: ZR
Caesar 03: YQ
Caesar 02: XP
Caesar 01: WO
Caesar 00: VN
```

3 BASIC

3.1 Consulting Log

Expected hours needed: 6

Date	*Hours Spent*	*Tasks / Accomplishments / Issues / Thoughts*
03/19/2021	1	Spent some time learning the syntax, and setting up the base of the code. Also started the encrypt but did not finish it yet.
03/20/2021	2	I really did not have much debugging to do with Visual Basic. It worked exactly how I thought I would and things just felt intuitive and natural. There was also a good amount of resources online to be able to figure out syntax and helpful functions.

3.2 Commentary

I personally liked Visual Basic the best, not only because it took me the least amount of time to complete, but because it felt intuitive as I was writing it. It translated really well in my head when I was writing it although I am not sure if that is just because I already wrote the code 2 other times and spent a lot more time debugging the other ones. Also I think what made it the most enjoyable was the amount of documentation online for this version. All the documentation was really helpful and easy to follow. I would definitely base my own language off of some of the syntax and functions in this language.

3.3 Source Code

```
Imports System
Module Test
        Function encrypt(testStr as string, shift as integer) as string
                DIM encryptedStr as string
                DIM e as integer

                for e = 0 to len(testStr)-1
                        DIM letter as char = testStr(e)
                        if((Asc(letter) >= 65) and (Asc(letter) <= 90)) then
                                if((Asc(letter) = 65) and (shift < 0)) then
                                        encryptedStr += Chr(Asc(letter) + 26 + shift)
                                else if((Asc(letter) = 90) and (shift > 0)) then
                                        encryptedStr += Chr(Asc(letter) - 26 + shift)
                                else
                                        encryptedStr += Chr(Asc(letter) + shift)
                                End if
                        Else
                                encryptedStr += letter
                        End if
                Next
```

```
21              encrypt = encryptedStr
22      End Function
23
24      Function decrypt(testStr as string, shift as integer) as string
25              DIM decryptedStr as string
26              DIM d as integer
27
28              for d = 0 to len(testStr)-1
29                      DIM letter as char = testStr(d)
30                      if((Asc(letter) >= 65) and (Asc(letter) <= 90)) then
31                              if((Asc(letter) = 65) and (-shift < 0)) then
32                                      decryptedStr += Chr(Asc(letter) + 26 - shift)
33                              else if((Asc(letter) = 90) and (-shift > 0)) then
34                                      decryptedStr += Chr(Asc(letter) - 26 - shift)
35                              else
36                                      decryptedStr += Chr(Asc(letter) - shift)
37                              End if
38                      Else
39                              decryptedStr += letter
40                      End if
41              Next
42              decrypt = decryptedStr
43      End Function
44
45      Function solve(testStr as string, maxShift as integer) as string
46              DIM solvedChrArr() as char = testStr.ToCharArray
47              DIM solvedStr as string
48              DIM shift as integer = -1
49              DIM j as integer
50              DIM s as integer
51
52              Console.WriteLine(vbTab & "Caesar 26: " + testStr)
53              for j = 0 to maxShift-1
54                      for s = 0 to len(testStr)-1
55                              DIM letter as char = solvedChrArr(s)
56                              if((Asc(letter) >= 65) and (Asc(letter) <= 90)) then
57                                      if((Asc(letter) = 65) and (shift < 0)) then
58                                              solvedChrArr(s) = Chr(Asc(letter) + 26 + shift)
59                                      else if((Asc(letter) = 90) and (shift > 0)) then
60                                              solvedChrArr(s) = Chr(Asc(letter) - 26 + shift)
61                                      else
62                                              solvedChrArr(s) = Chr(Asc(letter) + shift)
63                                      End if
64                              Else
65                                      solvedChrArr(s) = letter
66                              End if
67                      Next
68                      solvedStr = New String(solvedChrArr)
69                      DIM tempNum as integer = maxShift-j-1
70                      Console.WriteLine(vbTab & "Caesar " + tempNum.toString() + ": " + solvedStr)
71              Next
72              solve = solvedStr
73      End Function
74
75      Sub Main()
76              ' your code goes here
77              Console.WriteLine("Caesar Cipher" & vbLf)
78              DIM testStr1 as string = "IBM"
79      DIM testStr2 as string = "Hello World"
80      DIM testStr3 as string = "This is a test"
81      DIM testStr4 as string = "Hi my name is Robbie"
82      DIM testStr5 as string = "WandaVision"
83      DIM testStr6 as string = "Abed"
84      DIM testStr7 as string = "The Mandalorian"
85      DIM testStr8 as string = "Wow I learned BASIC I think"
```

```
86
87          DIM shiftAmount as integer = -1
88          DIM maxShiftAmount as integer = 26
89
90          DIM testStrArr = New String() {testStr1, testStr2, testStr3, testStr4, testStr5, testStr6, testStr7
91
92          DIM i as integer
93          for i = 0 to (testStrArr.length - 1)
94                  'Console.WriteLine(testStrArr(i))
95                  DIM tempStr as string = UCase(testStrArr(i))
96                  Console.WriteLine("Original String: " + tempStr)
97
98                  DIM encryptedTempStr as string = encrypt(tempStr, shiftAmount)
99                  Console.WriteLine("Encrypted String: " + encryptedTempStr)
100
101                 DIM decryptedTempStr as string = decrypt(encryptedTempStr, shiftAmount)
102                 Console.WriteLine("Decrypted String: " + decryptedTempStr)
103
104                 Console.WriteLine("Solve: ")
105                 solve(encryptedTempStr, maxShiftAmount)
106                 Console.WriteLine()
107         next
108         End Sub
109 End Module
```

3.4 Output

```
1  Caesar Cipher
2
3  Original String: IBM
4  Encrypted String: HAL
5  Decrypted String: IBM
6  Solve:
7          Caesar 26: HAL
8          Caesar 25: GZK
9          Caesar 24: FYJ
10         Caesar 23: EXI
11         Caesar 22: DWH
12         Caesar 21: CVG
13         Caesar 20: BUF
14         Caesar 19: ATE
15         Caesar 18: ZSD
16         Caesar 17: YRC
17         Caesar 16: XQB
18         Caesar 15: WPA
19         Caesar 14: VOZ
20         Caesar 13: UNY
21         Caesar 12: TMX
22         Caesar 11: SLW
23         Caesar 10: RKV
24         Caesar 9: QJU
25         Caesar 8: PIT
26         Caesar 7: OHS
27         Caesar 6: NGR
28         Caesar 5: MFQ
29         Caesar 4: LEP
30         Caesar 3: KDO
31         Caesar 2: JCN
32         Caesar 1: IBM
33         Caesar 0: HAL
34
35 Original String: HELLO WORLD
36 Encrypted String: GDKKN VNQKC
```

```
Decrypted String: HELLO WORLD
Solve:
        Caesar 26: GDKKN VNQKC
        Caesar 25: FCJJM UMPJB
        Caesar 24: EBIIL TLOIA
        Caesar 23: DAHHK SKNHZ
        Caesar 22: CZGGJ RJMGY
        Caesar 21: BYFFI QILFX
        Caesar 20: AXEEH PHKEW
        Caesar 19: ZWDDG OGJDV
        Caesar 18: YVCCF NFICU
        Caesar 17: XUBBE MEHBT
        Caesar 16: WTAAD LDGAS
        Caesar 15: VSZZC KCFZR
        Caesar 14: URYYB JBEYQ
        Caesar 13: TQXXA IADXP
        Caesar 12: SPWWZ HZCWO
        Caesar 11: ROVVY GYBVN
        Caesar 10: QNUUX FXAUM
        Caesar 9: PMTTW EWZTL
        Caesar 8: OLSSV DVYSK
        Caesar 7: NKRRU CUXRJ
        Caesar 6: MJQQT BTWQI
        Caesar 5: LIPPS ASVPH
        Caesar 4: KHOOR ZRUOG
        Caesar 3: JGNNQ YQTNF
        Caesar 2: IFMMP XPSME
        Caesar 1: HELLO WORLD
        Caesar 0: GDKKN VNQKC

Original String: THIS IS A TEST
Encrypted String: SGHR HR Z SDRS
Decrypted String: THIS IS A TEST
Solve:
        Caesar 26: SGHR HR Z SDRS
        Caesar 25: RFGQ GQ Y RCQR
        Caesar 24: QEFP FP X QBPQ
        Caesar 23: PDEO EO W PAOP
        Caesar 22: OCDN DN V OZNO
        Caesar 21: NBCM CM U NYMN
        Caesar 20: MABL BL T MXLM
        Caesar 19: LZAK AK S LWKL
        Caesar 18: KYZJ ZJ R KVJK
        Caesar 17: JXYI YI Q JUIJ
        Caesar 16: IWXH XH P ITHI
        Caesar 15: HVWG WG O HSGH
        Caesar 14: GUVF VF N GRFG
        Caesar 13: FTUE UE M FQEF
        Caesar 12: ESTD TD L EPDE
        Caesar 11: DRSC SC K DOCD
        Caesar 10: CQRB RB J CNBC
        Caesar 9: BPQA QA I BMAB
        Caesar 8: AOPZ PZ H ALZA
        Caesar 7: ZNOY OY G ZKYZ
        Caesar 6: YMNX NX F YJXY
        Caesar 5: XLMW MW E XIWX
        Caesar 4: WKLV LV D WHVW
        Caesar 3: VJKU KU C VGUV
        Caesar 2: UIJT JT B UFTU
        Caesar 1: THIS IS A TEST
        Caesar 0: SGHR HR Z SDRS

Original String: HI MY NAME IS ROBBIE
Encrypted String: GH LX MZLD HR QNAAHD
Decrypted String: HI MY NAME IS ROBBIE
```

```
Solve:
        Caesar 26: GH LX MZLD HR QNAAHD
        Caesar 25: FG KW LYKC GQ PMZZGC
        Caesar 24: EF JV KXJB FP OLYYFB
        Caesar 23: DE IU JWIA EO NKXXEA
        Caesar 22: CD HT IVHZ DN MJWWDZ
        Caesar 21: BC GS HUGY CM LIVVCY
        Caesar 20: AB FR GTFX BL KHUUBX
        Caesar 19: ZA EQ FSEW AK JGTTAW
        Caesar 18: YZ DP ERDV ZJ IFSSZV
        Caesar 17: XY CO DQCU YI HERRYU
        Caesar 16: WX BN CPBT XH GDQQXT
        Caesar 15: VW AM BOAS WG FCPPWS
        Caesar 14: UV ZL ANZR VF EBOOVR
        Caesar 13: TU YK ZMYQ UE DANNUQ
        Caesar 12: ST XJ YLXP TD CZMMTP
        Caesar 11: RS WI XKWO SC BYLLSO
        Caesar 10: QR VH WJVN RB AXKKRN
        Caesar 9: PQ UG VIUM QA ZWJJQM
        Caesar 8: OP TF UHTL PZ YVIIPL
        Caesar 7: NO SE TGSK OY XUHHOK
        Caesar 6: MN RD SFRJ NX WTGGNJ
        Caesar 5: LM QC REQI MW VSFFMI
        Caesar 4: KL PB QDPH LV UREELH
        Caesar 3: JK OA PCOG KU TQDDKG
        Caesar 2: IJ NZ OBNF JT SPCCJF
        Caesar 1: HI MY NAME IS ROBBIE
        Caesar 0: GH LX MZLD HR QNAAHD

Original String: WANDAVISION
Encrypted String: VZMCZUHRHNM
Decrypted String: WANDAVISION
Solve:
        Caesar 26: VZMCZUHRHNM
        Caesar 25: UYLBYTGQGML
        Caesar 24: TXKAXSFPFLK
        Caesar 23: SWJZWREOEKJ
        Caesar 22: RVIYVQDNDJI
        Caesar 21: QUHXUPCMCIH
        Caesar 20: PTGWTOBLBHG
        Caesar 19: OSFVSNAKAGF
        Caesar 18: NREURMZJZFE
        Caesar 17: MQDTQLYIYED
        Caesar 16: LPCSPKXHXDC
        Caesar 15: KOBROJWGWCB
        Caesar 14: JNAQNIVFVBA
        Caesar 13: IMZPMHUEUAZ
        Caesar 12: HLYOLGTDTZY
        Caesar 11: GKXNKFSCSYX
        Caesar 10: FJWMJERBRXW
        Caesar 9: EIVLIDQAQWV
        Caesar 8: DHUKHCPZPVU
        Caesar 7: CGTJGBOYOUT
        Caesar 6: BFSIFANXNTS
        Caesar 5: AERHEZMWMSR
        Caesar 4: ZDQGDYLVLRQ
        Caesar 3: YCPFCXKUKQP
        Caesar 2: XBOEBWJTJPO
        Caesar 1: WANDAVISION
        Caesar 0: VZMCZUHRHNM

Original String: ABED
Encrypted String: ZADC
Decrypted String: ABED
Solve:
```

```
        Caesar 26: ZADC
        Caesar 25: YZCB
        Caesar 24: XYBA
        Caesar 23: WXAZ
        Caesar 22: VWZY
        Caesar 21: UVYX
        Caesar 20: TUXW
        Caesar 19: STWV
        Caesar 18: RSVU
        Caesar 17: QRUT
        Caesar 16: PQTS
        Caesar 15: OPSR
        Caesar 14: NORQ
        Caesar 13: MNQP
        Caesar 12: LMPO
        Caesar 11: KLON
        Caesar 10: JKNM
        Caesar 9: IJML
        Caesar 8: HILK
        Caesar 7: GHKJ
        Caesar 6: FGJI
        Caesar 5: EFIH
        Caesar 4: DEHG
        Caesar 3: CDGF
        Caesar 2: BCFE
        Caesar 1: ABED
        Caesar 0: ZADC

Original String: THE MANDALORIAN
Encrypted String: SGD LZMCZKNQHZM
Decrypted String: THE MANDALORIAN
Solve:
        Caesar 26: SGD LZMCZKNQHZM
        Caesar 25: RFC KYLBYJMPGYL
        Caesar 24: QEB JXKAXILOFXK
        Caesar 23: PDA IWJZWHKNEWJ
        Caesar 22: OCZ HVIYVGJMDVI
        Caesar 21: NBY GUHXUFILCUH
        Caesar 20: MAX FTGWTEHKBTG
        Caesar 19: LZW ESFVSDGJASF
        Caesar 18: KYV DREURCFIZRE
        Caesar 17: JXU CQDTQBEHYQD
        Caesar 16: IWT BPCSPADGXPC
        Caesar 15: HVS AOBROZCFWOB
        Caesar 14: GUR ZNAQNYBEVNA
        Caesar 13: FTQ YMZPMXADUMZ
        Caesar 12: ESP XLYOLWZCTLY
        Caesar 11: DRO WKXNKVYBSKX
        Caesar 10: CQN VJWMJUXARJW
        Caesar 9: BPM UIVLITWZQIV
        Caesar 8: AOL THUKHSVYPHU
        Caesar 7: ZNK SGTJGRUXOGT
        Caesar 6: YMJ RFSIFQTWNFS
        Caesar 5: XLI QERHEPSVMER
        Caesar 4: WKH PDQGDORULDQ
        Caesar 3: VJG OCPFCNQTKCP
        Caesar 2: UIF NBOEBMPSJBO
        Caesar 1: THE MANDALORIAN
        Caesar 0: SGD LZMCZKNQHZM

Original String: WOW I LEARNED BASIC I THINK
Encrypted String: VNV H KDZQMDC AZRHB H SGHMJ
Decrypted String: WOW I LEARNED BASIC I THINK
Solve:
        Caesar 26: VNV H KDZQMDC AZRHB H SGHMJ
```

```
Caesar 25: UMU G JCYPLCB ZYQGA G RFGLI
Caesar 24: TLT F IBXOKBA YXPFZ F QEFKH
Caesar 23: SKS E HAWNJAZ XWOEY E PDEJG
Caesar 22: RJR D GZVMIZY WVNDX D OCDIF
Caesar 21: QIQ C FYULHYX VUMCW C NBCHE
Caesar 20: PHP B EXTKGXW UTLBV B MABGD
Caesar 19: OGO A DWSJFWV TSKAU A LZAFC
Caesar 18: NFN Z CVRIEVU SRJZT Z KYZEB
Caesar 17: MEM Y BUQHDUT RQIYS Y JXYDA
Caesar 16: LDL X ATPGCTS QPHXR X IWXCZ
Caesar 15: KCK W ZSOFBSR POGWQ W HVWBY
Caesar 14: JBJ V YRNEARQ ONFVP V GUVAX
Caesar 13: IAI U XQMDZQP NMEUO U FTUZW
Caesar 12: HZH T WPLCYPO MLDTN T ESTYV
Caesar 11: GYG S VOKBXON LKCSM S DRSXU
Caesar 10: FXF R UNJAWNM KJBRL R CQRWT
Caesar 9: EWE Q TMIZVML JIAQK Q BPQVS
Caesar 8: DVD P SLHYULK IHZPJ P AOPUR
Caesar 7: CUC O RKGXTKJ HGYOI O ZNOTQ
Caesar 6: BTB N QJFWSJI GFXNH N YMNSP
Caesar 5: ASA M PIEVRIH FEWMG M XLMRO
Caesar 4: ZRZ L OHDUQHG EDVLF L WKLQN
Caesar 3: YQY K NGCTPGF DCUKE K VJKPM
Caesar 2: XPX J MFBSOFE CBTJD J UIJOL
Caesar 1: WOW I LEARNED BASIC I THINK
Caesar 0: VNV H KDZQMDC AZRHB H SGHMJ
```

4 Pascal

4.1 Consulting Log

Expected hours needed: 8

Date	***Hours Spent***	***Tasks / Accomplishments / Issues / Thoughts***
03/15/2021	1	Learned syntax and started it based off of what I had done for Scala.
03/17/2021	3	I wrote out the encrypt function, however I am getting some weird errors that I can't discern. I finally got some of them to go away when I added parenthesis and found out how to properly type cast some variables.
03/18/2021	3	Spent a lot of time debugging the encrypt function and finally got it to work. It is still buggy though. I also wrote some of the decrypt function based off of the encrypt, however, both are buggy with certain letters not moving in the right direction. This is because the if statements are not working how I would like them to. "else if" and "else" syntax are giving me a really hard time.
03/19/2021	2.5	After a couple of hours of trying to figure it out, I finally figured out the BEGIN and END for the nested if statements that contained else and else if's with help from your email. The syntax of it was what threw me for a loop and I could not wrap my head around it for a while.

4.2 Commentary

Pascal was not at all bad to work with and it had some great concept things in the context and in functions. However, I thought some of the syntax was confusing especially with If, else if, and else statements. Everything else was fine, but ending these statements was confusing and did not work how I thought they did or should

have. The documentation for this online mainly was not helpful until I emailed you which cleared up how to group/end the statements.

4.3 Source Code

```
program ideone;
(*define encrypt function*)
function encrypt(testStr: string; shift: integer): string;
var
        encryptedChrArr : array of char;
        letter : char;
        encryptedStr : string;
        x, e : integer;
BEGIN
    encryptedStr := '';
        SetLength(encryptedChrArr, length(testStr));

        for x:=0 to length(testStr)-1 do
        BEGIN
                encryptedChrArr[x] := testStr[x+1]; (*Add each character of string to a char array*)
                (*writeln(encryptedChrArr[x]);*)
        END; {for x}

        for e:=0 to length(encryptedChrArr)-1 do
        BEGIN
                letter := encryptedChrArr[e];
                (*writeln(letter);*) (*check letters*)
                if((Integer(letter) >= 65) and (Integer(letter) <= 90)) then BEGIN(*A to Z*)
                        if((Integer(letter) = 65) and (shift < 0)) then BEGIN(*If A and shift is negative,
                                encryptedChrArr[e] := Chr(Integer(letter) + 26 + shift);
                    END else if((Integer(letter) = 90) and (shift > 0)) then BEGIN(*If Z, and shift is posi
                                encryptedChrArr[e] := Chr(Integer(letter) - 26 + shift);
                    END else BEGIN(*Perform a normal shift*)
                                encryptedChrArr[e] := Chr(Integer(letter) + shift);
                            END;
                END else BEGIN (*Other Characters*)
                            encryptedChrArr[e] := letter;
                END; {if}
                (*turn character array into new string*)
                encryptedStr := encryptedStr + encryptedChrArr[e];
        END; {for e}
        encrypt := encryptedStr;
END; {function encrypt}

(*define decrypt function*)
function decrypt(testStr: string; shift: integer): string;
var
        decryptedChrArr : array of char;
        letter : char;
        decryptedStr : string;
        y, d : integer;

BEGIN
    decryptedStr := '';
        SetLength(decryptedChrArr, length(testStr));

        for y:=0 to length(testStr)-1 do
        BEGIN
                decryptedChrArr[y] := testStr[y+1]; (*Add each character of string to a char array*)
                (*writeln(decryptedChrArr[x]);*)
        END; {for y}

        for d:=0 to length(decryptedChrArr)-1 do
        BEGIN
```

```
60                letter := decryptedChrArr[d];
61                (*writeln(letter);*) (*check letters*)
62                if((Integer(letter) >= 65) and (Integer(letter) <= 90)) then BEGIN(*A to Z*)
63                        if((Integer(letter) = 65) and (-shift < 0)) then BEGIN(*If A and shift is negative,
64                                decryptedChrArr[d] := Chr(Integer(letter) + 26 - shift);
65                    END else if((Integer(letter) = 90) and (-shift > 0)) then BEGIN(*If Z, and shift is pos
66                                decryptedChrArr[d] := Chr(Integer(letter) - 26 - shift);
67                    END else BEGIN(*Perform a normal shift*)
68                                decryptedChrArr[d] := Chr(Integer(letter) - shift);
69                        END;
70                END else BEGIN (*Other Characters*)
71                        decryptedChrArr[d] := letter;
72                END; {if}
73                (*turn character array into new string*)
74                decryptedStr := decryptedStr + decryptedChrArr[d];
75        END; {for d}
76        decrypt := decryptedStr;
77 END; {function decrpyt}
78
79
80 (*define solve function*)
81 function solve(testStr: string; maxShift: integer) : string;
82 var
83        solvedChrArr : array of char;
84        letter : char;
85        solvedStr : string;
86        shift : integer;
87        tempNum : integer;
88        n, j, s : integer;
89
90 BEGIN
91        writeln(#9'Caesar 26: ' + testStr);
92        solvedStr := '';
93        shift := -1;
94        SetLength(solvedChrArr, length(testStr));
95
96        for n:=0 to length(testStr)-1 do
97        BEGIN
98                solvedChrArr[n] := testStr[n+1]; (*Add each character of string to a char array*)
99                (*writeln(decryptedChrArr[x]);*)
100       END; {for n}
101
102       for j:=0 to maxShift-1 do
103       BEGIN
104               solvedStr := '';
105               for s:=0 to length(solvedChrArr)-1 do
106               BEGIN
107                       letter := solvedChrArr[s];
108                       (*writeln(letter);*) (*check letters*)
109                       if((Integer(letter) >= 65) and (Integer(letter) <= 90)) then BEGIN(*A to Z*)
110                               if((Integer(letter) = 65) and (shift < 0)) then BEGIN(*If A and shift is ne
111                                       solvedChrArr[s] := Chr(Integer(letter) + 26 + shift);
112                           END else if((Integer(letter) = 90) and (shift > 0)) then BEGIN(*If Z, and shift
113                                       solvedChrArr[s] := Chr(Integer(letter) - 26 + shift);
114                           END else BEGIN(*Perform a normal shift*)
115                                       solvedChrArr[s] := Chr(Integer(letter) + shift);
116                               END;
117                       END else BEGIN (*Other Characters*)
118                               solvedChrArr[s] := letter;
119                       END; {if}
120                       (*turn character array into new string*)
121                       solvedStr := solvedStr + solvedChrArr[s];
122               END; {for s}
123               tempNum := maxShift-j-1;
124               writeln(#9'Caesar ', tempNum,  ': ' + solvedStr);
```

```
        END; {for j}
        solve := solvedStr;
END; {function solve}

var
        testStr1, testStr2, testStr3, testStr4, testStr5, testStr6, testStr7, testStr8 : string;
        shiftAmount, maxShiftAmount : integer;
        testStrArr : array[0..7] of string;

        tempStr, encryptedTempStr, decryptedTempStr, solvedTempStr : string;
        i : integer;

BEGIN
        writeln('Caesar Cipher');
        writeln();
        testStr1 := 'IBM';
        testStr2 := 'Hello World';
        testStr3 := 'This is a test';
        testStr4 := 'Hi my name is Robbie';
        testStr5 := 'WandaVision';
        testStr6 := 'Abed';
        testStr7 := 'The Mandalorian';
        testStr8 := 'Wow I learned Pascal I think';

        shiftAmount := -1;
        maxShiftAmount := 26;

        testStrArr[0] := testStr1; testStrArr[1] := testStr2;
        testStrArr[2] := testStr3; testStrArr[3] := testStr4;
        testStrArr[4] := testStr5; testStrArr[5] := testStr6;
        testStrArr[6] := testStr7; testStrArr[7] := testStr8;

        for i:=0 to length(testStrArr)-1 do
        BEGIN
                tempStr := UpCase(testStrArr[i]); (*Change all letters to uppercase*)
                writeln('Original String: ' + tempStr);

                encryptedTempStr := encrypt(tempStr, shiftAmount);
                writeln('Encrypted String: ' + encryptedTempStr);

                decryptedTempStr := decrypt(encryptedTempStr, shiftAmount);
                writeln('Decrypted String: ' + decryptedTempStr);

                writeln('Solve: ');
                solvedTempStr := solve(encryptedTempStr, maxShiftAmount);
                writeln();
        END; {for i}
END. {program ideone}
```

4.4 Output

```
Caesar Cipher

Original String: IBM
Encrypted String: HAL
Decrypted String: IBM
Solve:
        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

Original String: HELLO WORLD
Encrypted String: GDKKN VNQKC
Decrypted String: HELLO WORLD
Solve:
        Caesar 26: GDKKN VNQKC
        Caesar 25: FCJJM UMPJB
        Caesar 24: EBIIL TLOIA
        Caesar 23: DAHHK SKNHZ
        Caesar 22: CZGGJ RJMGY
        Caesar 21: BYFFI QILFX
        Caesar 20: AXEEH PHKEW
        Caesar 19: ZWDDG OGJDV
        Caesar 18: YVCCF NFICU
        Caesar 17: XUBBE MEHBT
        Caesar 16: WTAAD LDGAS
        Caesar 15: VSZZC KCFZR
        Caesar 14: URYYB JBEYQ
        Caesar 13: TQXXA IADXP
        Caesar 12: SPWWZ HZCWO
        Caesar 11: ROVVY GYBVN
        Caesar 10: QNUUX FXAUM
        Caesar 9: PMTTW EWZTL
        Caesar 8: OLSSV DVYSK
        Caesar 7: NKRRU CUXRJ
        Caesar 6: MJQQT BTWQI
        Caesar 5: LIPPS ASVPH
        Caesar 4: KHOOR ZRUOG
        Caesar 3: JGNNQ YQTNF
        Caesar 2: IFMMP XPSME
        Caesar 1: HELLO WORLD
        Caesar 0: GDKKN VNQKC

Original String: THIS IS A TEST
Encrypted String: SGHR HR Z SDRS
Decrypted String: THIS IS A TEST
Solve:
        Caesar 26: SGHR HR Z SDRS
        Caesar 25: RFGQ GQ Y RCQR
        Caesar 24: QEFP FP X QBPQ
        Caesar 23: PDEO EO W PAOP
        Caesar 22: OCDN DN V OZNO
        Caesar 21: NBCM CM U NYMN
        Caesar 20: MABL BL T MXLM
```

```
        Caesar 19: LZAK AK S LWKL
        Caesar 18: KYZJ ZJ R KVJK
        Caesar 17: JXYI YI Q JUIJ
        Caesar 16: IWXH XH P ITHI
        Caesar 15: HVWG WG O HSGH
        Caesar 14: GUVF VF N GRFG
        Caesar 13: FTUE UE M FQEF
        Caesar 12: ESTD TD L EPDE
        Caesar 11: DRSC SC K DOCD
        Caesar 10: CQRB RB J CNBC
        Caesar 9: BPQA QA I BMAB
        Caesar 8: AOPZ PZ H ALZA
        Caesar 7: ZNOY OY G ZKYZ
        Caesar 6: YMNX NX F YJXY
        Caesar 5: XLMW MW E XIWX
        Caesar 4: WKLV LV D WHVW
        Caesar 3: VJKU KU C VGUV
        Caesar 2: UIJT JT B UFTU
        Caesar 1: THIS IS A TEST
        Caesar 0: SGHR HR Z SDRS

Original String: HI MY NAME IS ROBBIE
Encrypted String: GH LX MZLD HR QNAAHD
Decrypted String: HI MY NAME IS ROBBIE
Solve:
        Caesar 26: GH LX MZLD HR QNAAHD
        Caesar 25: FG KW LYKC GQ PMZZGC
        Caesar 24: EF JV KXJB FP OLYYFB
        Caesar 23: DE IU JWIA EO NKXXEA
        Caesar 22: CD HT IVHZ DN MJWWDZ
        Caesar 21: BC GS HUGY CM LIVVCY
        Caesar 20: AB FR GTFX BL KHUUBX
        Caesar 19: ZA EQ FSEW AK JGTTAW
        Caesar 18: YZ DP ERDV ZJ IFSSZV
        Caesar 17: XY CO DQCU YI HERRYU
        Caesar 16: WX BN CPBT XH GDQQXT
        Caesar 15: VW AM BOAS WG FCPPWS
        Caesar 14: UV ZL ANZR VF EBOOVR
        Caesar 13: TU YK ZMYQ UE DANNUQ
        Caesar 12: ST XJ YLXP TD CZMMTP
        Caesar 11: RS WI XKWO SC BYLLSO
        Caesar 10: QR VH WJVN RB AXKKRN
        Caesar 9: PQ UG VIUM QA ZWJJQM
        Caesar 8: OP TF UHTL PZ YVIIPL
        Caesar 7: NO SE TGSK OY XUHHOK
        Caesar 6: MN RD SFRJ NX WTGGNJ
        Caesar 5: LM QC REQI MW VSFFMI
        Caesar 4: KL PB QDPH LV UREELH
        Caesar 3: JK OA PCOG KU TQDDKG
        Caesar 2: IJ NZ OBNF JT SPCCJF
        Caesar 1: HI MY NAME IS ROBBIE
        Caesar 0: GH LX MZLD HR QNAAHD

Original String: WANDAVISION
Encrypted String: VZMCZUHRHNM
Decrypted String: WANDAVISION
Solve:
        Caesar 26: VZMCZUHRHNM
        Caesar 25: UYLBYTGQGML
        Caesar 24: TXKAXSFPFLK
        Caesar 23: SWJZWREOEKJ
        Caesar 22: RVIYVQDNDJI
        Caesar 21: QUHXUPCMCIH
        Caesar 20: PTGWTOBLBHG
        Caesar 19: OSFVSNAKAGF
```

```
        Caesar 18: NREURMZJZFE
        Caesar 17: MQDTQLYIYED
        Caesar 16: LPCSPKXHXDC
        Caesar 15: KOBROJWGWCB
        Caesar 14: JNAQNIVFVBA
        Caesar 13: IMZPMHUEUAZ
        Caesar 12: HLYOLGTDTZY
        Caesar 11: GKXNKFSCSYX
        Caesar 10: FJWMJERBRXW
        Caesar 9: EIVLIDQAQWV
        Caesar 8: DHUKHCPZPVU
        Caesar 7: CGTJGBOYOUT
        Caesar 6: BFSIFANXNTS
        Caesar 5: AERHEZMWMSR
        Caesar 4: ZDQGDYLVLRQ
        Caesar 3: YCPFCXKUKQP
        Caesar 2: XBOEBWJTJPO
        Caesar 1: WANDAVISION
        Caesar 0: VZMCZUHRHNM

Original String: ABED
Encrypted String: ZADC
Decrypted String: ABED
Solve:
        Caesar 26: ZADC
        Caesar 25: YZCB
        Caesar 24: XYBA
        Caesar 23: WXAZ
        Caesar 22: VWZY
        Caesar 21: UVYX
        Caesar 20: TUXW
        Caesar 19: STWV
        Caesar 18: RSVU
        Caesar 17: QRUT
        Caesar 16: PQTS
        Caesar 15: OPSR
        Caesar 14: NORQ
        Caesar 13: MNQP
        Caesar 12: LMPO
        Caesar 11: KLON
        Caesar 10: JKNM
        Caesar 9: IJML
        Caesar 8: HILK
        Caesar 7: GHKJ
        Caesar 6: FGJI
        Caesar 5: EFIH
        Caesar 4: DEHG
        Caesar 3: CDGF
        Caesar 2: BCFE
        Caesar 1: ABED
        Caesar 0: ZADC

Original String: THE MANDALORIAN
Encrypted String: SGD LZMCZKNQHZM
Decrypted String: THE MANDALORIAN
Solve:
        Caesar 26: SGD LZMCZKNQHZM
        Caesar 25: RFC KYLBYJMPGYL
        Caesar 24: QEB JXKAXILOFXK
        Caesar 23: PDA IWJZWHKNEWJ
        Caesar 22: OCZ HVIYVGJMDVI
        Caesar 21: NBY GUHXUFILCUH
        Caesar 20: MAX FTGWTEHKBTG
        Caesar 19: LZW ESFVSDGJASF
        Caesar 18: KYV DREURCFIZRE
```

```
        Caesar 17: JXU CQDTQBEHYQD
        Caesar 16: IWT BPCSPADGXPC
        Caesar 15: HVS AOBROZCFWOB
        Caesar 14: GUR ZNAQNYBEVNA
        Caesar 13: FTQ YMZPMXADUMZ
        Caesar 12: ESP XLYOLWZCTLY
        Caesar 11: DRO WKXNKVYBSKX
        Caesar 10: CQN VJWMJUXARJW
        Caesar 9: BPM UIVLITWZQIV
        Caesar 8: AOL THUKHSVYPHU
        Caesar 7: ZNK SGTJGRUXOGT
        Caesar 6: YMJ RFSIFQTWNFS
        Caesar 5: XLI QERHEPSVMER
        Caesar 4: WKH PDQGDORULDQ
        Caesar 3: VJG OCPFCNQTKCP
        Caesar 2: UIF NBOEBMPSJBO
        Caesar 1: THE MANDALORIAN
        Caesar 0: SGD LZMCZKNQHZM

Original String: WOW I LEARNED PASCAL I THINK
Encrypted String: VNV H KDZQMDC OZRBZK H SGHMJ
Decrypted String: WOW I LEARNED PASCAL I THINK
Solve:
        Caesar 26: VNV H KDZQMDC OZRBZK H SGHMJ
        Caesar 25: UMU G JCYPLCB NYQAYJ G RFGLI
        Caesar 24: TLT F IBXOKBA MXPZXI F QEFKH
        Caesar 23: SKS E HAWNJAZ LWOYWH E PDEJG
        Caesar 22: RJR D GZVMIZY KVNXVG D OCDIF
        Caesar 21: QIQ C FYULHYX JUMWUF C NBCHE
        Caesar 20: PHP B EXTKGXW ITLVTE B MABGD
        Caesar 19: OGO A DWSJFWV HSKUSD A LZAFC
        Caesar 18: NFN Z CVRIEVU GRJTRC Z KYZEB
        Caesar 17: MEM Y BUQHDUT FQISQB Y JXYDA
        Caesar 16: LDL X ATPGCTS EPHRPA X IWXCZ
        Caesar 15: KCK W ZSOFBSR DOGQOZ W HVWBY
        Caesar 14: JBJ V YRNEARQ CNFPNY V GUVAX
        Caesar 13: IAI U XQMDZQP BMEOMX U FTUZW
        Caesar 12: HZH T WPLCYPO ALDNLW T ESTYV
        Caesar 11: GYG S VOKBXON ZKCMKV S DRSXU
        Caesar 10: FXF R UNJAWNM YJBLJU R CQRWT
        Caesar 9: EWE Q TMIZVML XIAKIT Q BPQVS
        Caesar 8: DVD P SLHYULK WHZJHS P AOPUR
        Caesar 7: CUC O RKGXTKJ VGYIGR O ZNOTQ
        Caesar 6: BTB N QJFWSJI UFXHFQ N YMNSP
        Caesar 5: ASA M PIEVRIH TEWGEP M XLMRO
        Caesar 4: ZRZ L OHDUQHG SDVFDO L WKLQN
        Caesar 3: YQY K NGCTPGF RCUECN K VJKPM
        Caesar 2: XPX J MFBSOFE QBTDBM J UIJOL
        Caesar 1: WOW I LEARNED PASCAL I THINK
        Caesar 0: VNV H KDZQMDC OZRBZK H SGHMJ
```

5 Scala

5.1 Consulting Log

Expected hours needed: 6

Date	*Hours Spent*	*Tasks / Accomplishments / Issues / Thoughts*
03/10/2021	0.5	Time mostly spent learning the basics and learning the syntax.
03/11/2021	1	I mostly just spent a lot of time debugging.
03/12/2021	1	a lot more debugging with some more research and learning. My biggest issue between yesterday and today has just been with replacing the characters in the character array with a new value. So far I have only been able to get spaces to print out rather than what I actually thought it would. Mostly syntax errors but some conceptually errors on what is going on.
03/14	3	I FIGURED IT OUT. I did not work on it for terribly long before I finally got it to work today. Once I got the encryption done, decryption took significantly less time and the solve function took even less time with some room for me to make it look a little nicer. Getting that syntax and getting myself to write code that will do what I know I conceptually need to do was a big relief for me and it made my "late" night all worth it. I also ended up switching back to ideone.com because codingground was not working/ connecting to their servers.

5.2 Commentary

Scala felt very close to Java and other languages that I'm more used to, which I know was kind of the point. That made it easier for me to get that baseline of code down and figure out the logistics of writing the program without as many syntactical hiccups. This did not take me too long to write, or at least not as many headaches like Fortran and COBOL. Scala was enjoyable and I can see myself taking ideas from it and using it again in the future.

5.3 Source Code

```
object Main {
    def main(args: Array[String]) {
        println("Caesar Cipher\n")
        val testStr1 = "IBM"
        val testStr2 = "Hello World"
        val testStr3 = "This is a test"
        val testStr4 = "Hi my name is Robbie"
        val testStr5 = "WandaVision"
        val testStr6 = "Abed"
        val testStr7 = "The Mandalorian"
        val testStr8 = "Wow I learned Scala I think"

        val shiftAmount = -1
            val maxShiftAmount = 26

            var testStrArr = Array(testStr1, testStr2, testStr3, testStr4, testStr5, testStr6, testStr7, te

            for(i <- testStrArr) {
                var tempStr = i.toUpperCase()
                println("Original String: " + tempStr)

                var encryptedTempStr = encrypt(tempStr, shiftAmount)
                println("Encrypted String: " + encryptedTempStr)

                var decryptedTempStr = decrypt(encryptedTempStr, shiftAmount)
                println("Decrypted String: " + decryptedTempStr)

                println("Solve: ")
                solve(encryptedTempStr, maxShiftAmount)
```

```
            println()
        }
    }
    def encrypt(testStr:String, shift:Int) : String /*Array[Char]*/= {
        //var encryptedStr = ""
        var encryptedChrArr = testStr.toCharArray()
        var e = 0

        for(e <- 0 until testStr.length()) {
            var letter = encryptedChrArr(e) //testStr.charAt(e)
            //println(letter)
            if(letter.toInt >= 65 && letter.toInt <= 90) { //A to Z
                if(letter.toInt == 65 && shift < 0) { //If A and shift is negative, then loop around to Z
                    encryptedChrArr(e) = (letter.toInt + 26 + shift).toChar
                } else if(letter.toInt == 90 && shift > 0) { //If Z, and shift is positive, then loop aroun
                    encryptedChrArr(e) = (letter.toInt - 26 + shift).toChar
                } else { // Else perform a normal shift
                    encryptedChrArr(e) = (letter.toInt + shift).toChar
                    //encryptedStr.concat(letter.toString)
                    //encryptedChrArr(e) = (testStr.charAt(e).toInt + shift)
                    //println(letter)
                }
            }
            else { //Other Characters
                //Other Characters should not be changed in Caesar Cipher
                encryptedChrArr(e) = letter
                //println(letter)
            }
        }
        var encryptedStr = encryptedChrArr.mkString("")
        return encryptedStr
    }
    def decrypt(testStr:String, shift:Int) : String = {
        var decryptedChrArr = testStr.toCharArray()
        var d = 0

        for(d <- 0 until testStr.length()) {
            var letter = decryptedChrArr(d)
            //shift = -shift//Switch sign of shift
            //println(letter)
            if(letter.toInt >= 65 && letter.toInt <= 90) { //A to Z
                if(letter.toInt == 65 && -shift < 0) { //If A and shift is negative, then loop around to Z
                    decryptedChrArr(d) = (letter.toInt + 26 - shift).toChar
                } else if(letter.toInt == 90 && -shift > 0) { //If Z, and shift is positive, then loop arou
                    decryptedChrArr(d) = (letter.toInt - 26 - shift).toChar
                } else { // Else perform a normal shift
                    decryptedChrArr(d) = (letter.toInt - shift).toChar
                    //println(letter)
                }
            }
            else { //Other Characters
                //Other Characters should not be changed in Caesar Cipher
                decryptedChrArr(d) = letter
                //println(letter)
            }
        }
        var decryptedStr = decryptedChrArr.mkString("")
        return decryptedStr
    }
    def solve(testStr:String, maxShift:Int) {
        var solvedChrArr = testStr.toCharArray()
        val shift = -1
        var j = 0
        var s = 0

```

```
        println("\tCaesar " + (26).toString + ": " + testStr)
        for(j <- 0 until maxShift) {
                for(s <- 0 until testStr.length()) {
                    var letter = solvedChrArr(s)
                    //println(letter)
                    if(letter.toInt >= 65 && letter.toInt <= 90) { //A to Z
                        if(letter.toInt == 65 && shift < 0) { //If A and shift is negative, then loop aroun
                            solvedChrArr(s) = (letter.toInt + 26 + shift).toChar
                        } else if(letter.toInt == 90 && shift > 0) { //If Z, and shift is positive, then lo
                            solvedChrArr(s) = (letter.toInt - 26 + shift).toChar
                        } else { // Else perform a normal shift
                            solvedChrArr(s) = (letter.toInt + shift).toChar
                            //println(letter)
                        }
                    }
                    else { //Other Characters
                        solvedChrArr(s) = letter
                        //println(letter)
                    }
                }
                var solvedStr = solvedChrArr.mkString("")
                println("\tCaesar " + (maxShift-j-1).toString + ": " + solvedStr)
        }
    }
}
```

5.4 Output

```
Caesar Cipher

Original String: IBM
Encrypted String: HAL
Decrypted String: IBM
Solve:
        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

Original String: HELLO WORLD
```

```
Encrypted String: GDKKN VNQKC
Decrypted String: HELLO WORLD
Solve:
        Caesar 26: GDKKN VNQKC
        Caesar 25: FCJJM UMPJB
        Caesar 24: EBIIL TLOIA
        Caesar 23: DAHHK SKNHZ
        Caesar 22: CZGGJ RJMGY
        Caesar 21: BYFFI QILFX
        Caesar 20: AXEEH PHKEW
        Caesar 19: ZWDDG OGJDV
        Caesar 18: YVCCF NFICU
        Caesar 17: XUBBE MEHBT
        Caesar 16: WTAAD LDGAS
        Caesar 15: VSZZC KCFZR
        Caesar 14: URYYB JBEYQ
        Caesar 13: TQXXA IADXP
        Caesar 12: SPWWZ HZCWO
        Caesar 11: ROVVY GYBVN
        Caesar 10: QNUUX FXAUM
        Caesar 9: PMTTW EWZTL
        Caesar 8: OLSSV DVYSK
        Caesar 7: NKRRU CUXRJ
        Caesar 6: MJQQT BTWQI
        Caesar 5: LIPPS ASVPH
        Caesar 4: KHOOR ZRUOG
        Caesar 3: JGNNQ YQTNF
        Caesar 2: IFMMP XPSME
        Caesar 1: HELLO WORLD
        Caesar 0: GDKKN VNQKC

Original String: THIS IS A TEST
Encrypted String: SGHR HR Z SDRS
Decrypted String: THIS IS A TEST
Solve:
        Caesar 26: SGHR HR Z SDRS
        Caesar 25: RFGQ GQ Y RCQR
        Caesar 24: QEFP FP X QBPQ
        Caesar 23: PDEO EO W PAOP
        Caesar 22: OCDN DN V OZNO
        Caesar 21: NBCM CM U NYMN
        Caesar 20: MABL BL T MXLM
        Caesar 19: LZAK AK S LWKL
        Caesar 18: KYZJ ZJ R KVJK
        Caesar 17: JXYI YI Q JUIJ
        Caesar 16: IWXH XH P ITHI
        Caesar 15: HVWG WG O HSGH
        Caesar 14: GUVF VF N GRFG
        Caesar 13: FTUE UE M FQEF
        Caesar 12: ESTD TD L EPDE
        Caesar 11: DRSC SC K DOCD
        Caesar 10: CQRB RB J CNBC
        Caesar 9: BPQA QA I BMAB
        Caesar 8: AOPZ PZ H ALZA
        Caesar 7: ZNOY OY G ZKYZ
        Caesar 6: YMNX NX F YJXY
        Caesar 5: XLMW MW E XIWX
        Caesar 4: WKLV LV D WHVW
        Caesar 3: VJKU KU C VGUV
        Caesar 2: UIJT JT B UFTU
        Caesar 1: THIS IS A TEST
        Caesar 0: SGHR HR Z SDRS

Original String: HI MY NAME IS ROBBIE
Encrypted String: GH LX MZLD HR QNAAHD
```

```
Decrypted String: HI MY NAME IS ROBBIE
Solve:
        Caesar 26: GH LX MZLD HR QNAAHD
        Caesar 25: FG KW LYKC GQ PMZZGC
        Caesar 24: EF JV KXJB FP OLYYFB
        Caesar 23: DE IU JWIA EO NKXXEA
        Caesar 22: CD HT IVHZ DN MJWWDZ
        Caesar 21: BC GS HUGY CM LIVVCY
        Caesar 20: AB FR GTFX BL KHUUBX
        Caesar 19: ZA EQ FSEW AK JGTTAW
        Caesar 18: YZ DP ERDV ZJ IFSSZV
        Caesar 17: XY CO DQCU YI HERRYU
        Caesar 16: WX BN CPBT XH GDQQXT
        Caesar 15: VW AM BOAS WG FCPPWS
        Caesar 14: UV ZL ANZR VF EBOOVR
        Caesar 13: TU YK ZMYQ UE DANNUQ
        Caesar 12: ST XJ YLXP TD CZMMTP
        Caesar 11: RS WI XKWO SC BYLLSO
        Caesar 10: QR VH WJVN RB AXKKRN
        Caesar 9: PQ UG VIUM QA ZWJJQM
        Caesar 8: OP TF UHTL PZ YVIIPL
        Caesar 7: NO SE TGSK OY XUHHOK
        Caesar 6: MN RD SFRJ NX WTGGNJ
        Caesar 5: LM QC REQI MW VSFFMI
        Caesar 4: KL PB QDPH LV UREELH
        Caesar 3: JK OA PCOG KU TQDDKG
        Caesar 2: IJ NZ OBNF JT SPCCJF
        Caesar 1: HI MY NAME IS ROBBIE
        Caesar 0: GH LX MZLD HR QNAAHD

Original String: WANDAVISION
Encrypted String: VZMCZUHRHNM
Decrypted String: WANDAVISION
Solve:
        Caesar 26: VZMCZUHRHNM
        Caesar 25: UYLBYTGQGML
        Caesar 24: TXKAXSFPFLK
        Caesar 23: SWJZWREOEKJ
        Caesar 22: RVIYVQDNDJI
        Caesar 21: QUHXUPCMCIH
        Caesar 20: PTGWTOBLBHG
        Caesar 19: OSFVSNAKAGF
        Caesar 18: NREURMZJZFE
        Caesar 17: MQDTQLYIYED
        Caesar 16: LPCSPKXHXDC
        Caesar 15: KOBROJWGWCB
        Caesar 14: JNAQNIVFVBA
        Caesar 13: IMZPMHUEUAZ
        Caesar 12: HLYOLGTDTZY
        Caesar 11: GKXNKFSCSYX
        Caesar 10: FJWMJERBRXW
        Caesar 9: EIVLIDQAQWV
        Caesar 8: DHUKHCPZPVU
        Caesar 7: CGTJGBOYOUT
        Caesar 6: BFSIFANXNTS
        Caesar 5: AERHEZMWMSR
        Caesar 4: ZDQGDYLVLRQ
        Caesar 3: YCPFCXKUKQP
        Caesar 2: XBOEBWJTJPO
        Caesar 1: WANDAVISION
        Caesar 0: VZMCZUHRHNM

Original String: ABED
Encrypted String: ZADC
Decrypted String: ABED
```

```
Solve:
        Caesar 26: ZADC
        Caesar 25: YZCB
        Caesar 24: XYBA
        Caesar 23: WXAZ
        Caesar 22: VWZY
        Caesar 21: UVYX
        Caesar 20: TUXW
        Caesar 19: STWV
        Caesar 18: RSVU
        Caesar 17: QRUT
        Caesar 16: PQTS
        Caesar 15: OPSR
        Caesar 14: NORQ
        Caesar 13: MNQP
        Caesar 12: LMPO
        Caesar 11: KLON
        Caesar 10: JKNM
        Caesar 9: IJML
        Caesar 8: HILK
        Caesar 7: GHKJ
        Caesar 6: FGJI
        Caesar 5: EFIH
        Caesar 4: DEHG
        Caesar 3: CDGF
        Caesar 2: BCFE
        Caesar 1: ABED
        Caesar 0: ZADC

Original String: THE MANDALORIAN
Encrypted String: SGD LZMCZKNQHZM
Decrypted String: THE MANDALORIAN
Solve:
        Caesar 26: SGD LZMCZKNQHZM
        Caesar 25: RFC KYLBYJMPGYL
        Caesar 24: QEB JXKAXILOFXK
        Caesar 23: PDA IWJZWHKNEWJ
        Caesar 22: OCZ HVIYVGJMDVI
        Caesar 21: NBY GUHXUFILCUH
        Caesar 20: MAX FTGWTEHKBTG
        Caesar 19: LZW ESFVSDGJASF
        Caesar 18: KYV DREURCFIZRE
        Caesar 17: JXU CQDTQBEHYQD
        Caesar 16: IWT BPCSPADGXPC
        Caesar 15: HVS AOBROZCFWOB
        Caesar 14: GUR ZNAQNYBEVNA
        Caesar 13: FTQ YMZPMXADUMZ
        Caesar 12: ESP XLYOLWZCTLY
        Caesar 11: DRO WKXNKVYBSKX
        Caesar 10: CQN VJWMJUXARJW
        Caesar 9: BPM UIVLITWZQIV
        Caesar 8: AOL THUKHSVYPHU
        Caesar 7: ZNK SGTJGRUXOGT
        Caesar 6: YMJ RFSIFQTWNFS
        Caesar 5: XLI QERHEPSVMER
        Caesar 4: WKH PDQGDORULDQ
        Caesar 3: VJG OCPFCNQTKCP
        Caesar 2: UIF NBOEBMPSJBO
        Caesar 1: THE MANDALORIAN
        Caesar 0: SGD LZMCZKNQHZM

Original String: WOW I LEARNED SCALA I THINK
Encrypted String: VNV H KDZQMDC RBZKZ H SGHMJ
Decrypted String: WOW I LEARNED SCALA I THINK
Solve:
```

```
        Caesar 26: VNV H KDZQMDC RBZKZ H SGHMJ
        Caesar 25: UMU G JCYPLCB QAYJY G RFGLI
        Caesar 24: TLT F IBXOKBA PZXIX F QEFKH
        Caesar 23: SKS E HAWNJAZ OYWHW E PDEJG
        Caesar 22: RJR D GZVMIZY NXVGV D OCDIF
        Caesar 21: QIQ C FYULHYX MWUFU C NBCHE
        Caesar 20: PHP B EXTKGXW LVTET B MABGD
        Caesar 19: OGO A DWSJFWV KUSDS A LZAFC
        Caesar 18: NFN Z CVRIEVU JTRCR Z KYZEB
        Caesar 17: MEM Y BUQHDUT ISQBQ Y JXYDA
        Caesar 16: LDL X ATPGCTS HRPAP X IWXCZ
        Caesar 15: KCK W ZSOFBSR GQOZO W HVWBY
        Caesar 14: JBJ V YRNEARQ FPNYN V GUVAX
        Caesar 13: IAI U XQMDZQP EOMXM U FTUZW
        Caesar 12: HZH T WPLCYPO DNLWL T ESTYV
        Caesar 11: GYG S VOKBXON CMKVK S DRSXU
        Caesar 10: FXF R UNJAWNM BLJUJ R CQRWT
        Caesar 9: EWE Q TMIZVML AKITI Q BPQVS
        Caesar 8: DVD P SLHYULK ZJHSH P AOPUR
        Caesar 7: CUC O RKGXTKJ YIGRG O ZNOTQ
        Caesar 6: BTB N QJFWSJI XHFQF N YMNSP
        Caesar 5: ASA M PIEVRIH WGEPE M XLMRO
        Caesar 4: ZRZ L OHDUQHG VFDOD L WKLQN
        Caesar 3: YQY K NGCTPGF UECNC K VJKPM
        Caesar 2: XPX J MFBSOFE TDBMB J UIJOL
        Caesar 1: WOW I LEARNED SCALA I THINK
        Caesar 0: VNV H KDZQMDC RBZKZ H SGHMJ
```

6 Language Rankings

1. Visual BASIC
2. Scala
3. Pascal
4. Fortran
5. COBOL