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.

```
!Encrypt
2
  function encrypt(testStr, shift) result(encryptedStr)
           implicit none
           !local variables
5
           character(len = 30), intent(in) :: testStr
6
           integer, intent(in) :: shift
           !character, dimension(30) :: encryptedChrArr
           character :: letter
10
11
           integer :: asciiVal, e
12
           character(len = 30) :: encryptedStr
13
14
           encryptedStr = trim(testStr)
15
           eloop: do e = 1, len(encryptedStr)
16
               !if character from A to Z
17
                   letter = encryptedStr(e:e)
18
                    asciiVal = ichar(letter)
19
                    if((asciiVal \geq= 65) .and. (asciiVal <= 90)) then
20
                            if((asciiVal == 65) .and. (shift < 0)) then
21
                                     encryptedStr(e:e) = char(asciiVal + 26 + shift)
22
                            else if((asciiVal == 90) .and. (shift > 0)) then
23
                                     encryptedStr(e:e) = char(asciiVal - 26 + shift)
24
                            else
25
26
                                     encryptedStr(e:e) = char(asciiVal + shift)
                            end if
27
28
                    !if character from a to \boldsymbol{z}
29
                    else if((asciiVal \geq 97) .and. (asciiVal \leq 122)) then
                            if((asciiVal == 97) .and. (shift < 0)) then
30
                                     encryptedStr(e:e) = char(asciiVal + 26 + shift)
31
                            else if((asciiVal == 122) .and. (shift > 0)) then
32
33
                                     encryptedStr(e:e) = char(asciiVal - 26 + shift)
34
                            else
                                     encryptedStr(e:e) = char(asciiVal + shift)
35
36
                            end if
37
                        encryptedStr(e:e) = letter
38
                    end if
39
           end do eloop
40
  end function encrypt
41
42
43
  !Decrypt
  function decrypt(testStr, shift) result(decryptedStr)
44
45
           implicit none
46
47
           !local variables
           character(len = 30), intent(in) :: testStr
48
           integer, intent(in) :: shift
49
50
           character :: letter
51
           integer :: asciiVal, d
52
53
           character(len = 30) :: decryptedStr
54
55
           decryptedStr = trim(testStr)
56
           dloop: do d = 1, len(decryptedStr)
57
               !if character from A to {\bf Z}
58
                   letter = decryptedStr(d:d)
59
                    asciiVal = ichar(letter)
60
                    if((asciiVal >= 65) .and. (asciiVal <= 90)) then
61
62
                            if((asciiVal == 65) .and. (-shift < 0)) then
                                     decryptedStr(d:d) = char(asciiVal + 26 - shift)
63
```

```
else if((asciiVal == 90) .and. (-shift > 0)) then
64
                                      decryptedStr(d:d) = char(asciiVal - 26 - shift)
65
66
                             else
                                      decryptedStr(d:d) = char(asciiVal - shift)
67
                             end if
68
                    !if character from a to \boldsymbol{z}
69
                    else if((asciiVal >= 97) .and. (asciiVal <= 122)) then
70
                             if((asciiVal == 97) .and. (-shift < 0)) then
71
                                      decryptedStr(d:d) = char(asciiVal + 26 - shift)
72
                             else if((asciiVal == 122) .and. (-shift > 0)) then
73
                                      decryptedStr(d:d) = char(asciiVal - 26 - shift)
74
75
                             else
                                      decryptedStr(d:d) = char(asciiVal - shift)
76
77
                             end if
78
                    else
                         decryptedStr(d:d) = letter
79
                    end if
80
            end do dloop
81
   end function decrypt
82
83
84
   function solve(testStr, maxShift) result(solvedStr)
           implicit none
86
87
           !local variables
88
            character(len = 30), intent(in) :: testStr
89
90
            integer, intent(in) :: maxShift
91
92
            !character, dimension(30) :: solvedChrArr
            character :: letter
93
            integer :: shift, caesarNum
94
95
            integer :: asciiVal, j, s
96
            character(len = 30) :: solvedStr
97
98
            shift = -1
99
            solvedStr = trim(testStr)
100
           PRINT *, char(9), 'Caesar 26: ', solvedStr
101
102
            jloop: do j = 0, maxShift-1
            sloop: do s = 1, len(solvedStr)
103
                !if character from A to {\bf Z}
104
                    letter = solvedStr(s:s)
105
                    asciiVal = ichar(letter)
106
                    if((asciiVal >= 65) .and. (asciiVal <= 90)) then
107
                             if((asciiVal == 65) .and. (shift < 0)) then
108
                                      solvedStr(s:s) = char(asciiVal + 26 + shift)
109
                             else if((asciiVal == 90) .and. (shift > 0)) then
110
                                      solvedStr(s:s) = char(asciiVal - 26 + shift)
111
112
                             else
                                      solvedStr(s:s) = char(asciiVal + shift)
113
                             end if
114
                    !if character from a to z
115
                    else if((asciiVal \geq 97) .and. (asciiVal \leq 122)) then
116
                             if((asciiVal == 97) .and. (shift < 0)) then
117
                                      solvedStr(s:s) = char(asciiVal + 26 + shift)
118
119
                             else if((asciiVal == 122) .and. (shift > 0)) then
                                      solvedStr(s:s) = char(asciiVal - 26 + shift)
120
121
                                      solvedStr(s:s) = char(asciiVal + shift)
122
123
                             end if
124
                    else
                         solvedStr(s:s) = letter
125
                    end if
126
            end do sloop
127
           caesarNum = maxShift-j-1
128
```

```
print *, char(9), 'Caesar ', caesarNum, ': ', solvedStr
129
        end do jloop
130
131
   end function solve
132
   program MAIN
133
   !makes sure that variables "i,j,k,l,m,n" are not default to integer
134
135
            implicit none
136
   !Declare variables
137
138
            character(len = 30) :: encrypt !return type
            character(len = 30) :: decrypt !return type
139
            character(len = 30) :: solve
                                              !return type
140
141
            character(len = 30) :: testStr1, testStr2, testStr3, testStr4, testStr5, testStr6, testStr7, testSt
142
            character(len = 30), dimension(10) :: testStrArr(8) !one dimensional array
143
            \texttt{character(len = 30)} \ :: \ \texttt{tempStr} \ , \ \texttt{encryptedTempStr} \ , \ \texttt{decryptedTempStr} \ , \ \texttt{solvedTempStr} \ , \ \\
144
145
            integer :: shiftAmount
146
            integer :: maxShiftAmount
147
148
            integer :: i
149
   !Initialize variables
150
            testStr1 = 'IBM'
151
            testStr2 = 'Hello World'
152
            testStr3 = 'This is a test'
153
            testStr4 = 'Hi my name is Robbie'
154
            testStr5 = 'WandaVision'
155
            testStr6 = 'Abed'
156
            testStr7 = 'The Mandalorian'
157
            testStr8 = 'Wow I learned Fortran I think'
158
159
160
            shiftAmount = -1
            maxShiftAmount = 26
161
162
163
            testStrArr = (/ testStr1, testStr2, testStr3, testStr4, testStr5, testStr6, testStr7, testStr8 /)
164
165
            PRINT *, 'Caesar Cipher'
166
            PRINT *, ''
167
168
   !Do loop to encrypt, decrypt, and solve
169
            iloop: do i = 1, SIZE(testStrArr)
170
                     tempStr = (testStrArr(i))
PRINT *, 'Original String: ', tempStr
171
172
173
                     encryptedTempStr = encrypt(tempStr, shiftAmount)
174
                     PRINT *, 'Encrypted String: ', encryptedTempStr
175
                     decryptedTempStr = decrypt(encryptedTempStr, shiftAmount)
176
177
                     PRINT *, 'Decrpyted String: ', decryptedTempStr
178
                     PRINT *, 'Solve: '
179
                     solvedTempStr = solve(encryptedTempStr, maxShiftAmount)
180
                     PRINT *, ''
181
182
            end do iloop
183 end program MAIN
```

```
Caesar Cipher

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

```
Solve:
          Caesar 26: HAL
                            25 : GZK
          Caesar
                           24 : FYJ
          Caesar
9
                           23 : EXI
10
          Caesar
          Caesar
                           22 : DWH
11
          Caesar
                           21 : CVG
12
                           20 : BUF
13
          Caesar
          Caesar
                           19 : ATE
14
                           18 : ZSD
          Caesar
15
                          17 : YRC
          Caesar
16
17
          Caesar
                          16 : XQB
          Caesar
                           15 : WPA
18
19
          Caesar
                           14 : VOZ
                           13 : UNY
          Caesar
20
                          12 : TMX
          Caesar
21
                           11 : SLW
          Caesar
          Caesar
                           10 : RKV
23
          Caesar
                            9 : QJU
24
                           8 : PIT
          Caesar
25
                           7 : OHS
          Caesar
26
                           6 : NGR
27
          Caesar
                           5 : MFQ
4 : LEP
          Caesar
28
          Caesar
29
                           3 : KDO
30
          Caesar
          Caesar
                           2 : JCN
31
                            1 : IBM
32
          Caesar
          Caesar
                            O : HAL
33
34
   Original String: Hello World
35
   Encrypted String: Gdkkn Vnqkc
36
   Decrpyted String: Hello World
37
   Solve:
38
          Caesar 26: Gdkkn Vnqkc
39
          Caesar
                      25 : Fcjjm Umpjb
40
          Caesar
                           24 : Ebiil Tloia
41
                           23 : Dahhk Sknhz
          Caesar
42
          Caesar
                            22 : Czggj Rjmgy
43
                           21 : Byffi Qilfx
44
          Caesar
                          20 : Axeeh Phkew
          Caesar
45
          Caesar
                          19 : Zwddg Ogjdv
46
          Caesar
                          18 : Yvccf Nficu
47
          Caesar
                           17 : Xubbe Mehbt
48
                           16 : Wtaad Ldgas
          Caesar
49
          Caesar
                          15 : Vszzc Kcfzr
50
51
          Caesar
                          14 : Uryyb Jbeyq
          Caesar
                           13 : Tqxxa Iadxp
52
          Caesar
                           12 : Spwwz Hzcwo
53
          Caesar
                           11 : Rovvy Gybvn
54
          Caesar
                          10 : Qnuux Fxaum
55
                           9 : Pmttw Ewztl
          Caesar
                           8 : Olssv Dvysk
7 : Nkrru Cuxrj
          Caesar
57
          Caesar
58
                           6 : Mjqqt Btwqi
          Caesar
59
          Caesar
                           5 : Lipps Asvph
60
61
          Caesar
                            4 : Khoor Zruog
          Caesar
                            3 : Jgnnq Yqtnf
62
                            2 : Ifmmp Xpsme
          Caesar
63
                            1 : Hello World
          Caesar
64
          Caesar
                             0 : Gdkkn Vnqkc
66
   Original String: This is a test
67
   Encrypted String: Sghr hr z sdrs
69 Decryyted String: This is a test
70 Solve:
```

```
Caesar 26: Sghr hr z sdrs
71
           Caesar
                             25 : Rfgq gq y rcqr
72
73
           Caesar
                             24 : Qefp fp x qbpq
           Caesar
                             23 : Pdeo eo w paop
74
75
           Caesar
                             22 : Ocdn dn v ozno
                             21 : Nbcm cm u nymn
           Caesar
76
                             20 : Mabl bl t mxlm
77
           Caesar
                             19 : Lzak ak s lwkl
78
           Caesar
                             18 : Kyzj zj r kvjk
           Caesar
79
                             17 : Jxyi yi q juij
80
           Caesar
           Caesar
                             16 : Iwxh xh p ithi
81
           Caesar
                             15 : Hvwg wg o hsgh
82
                             14 : Guvf vf n grfg
           Caesar
83
84
           Caesar
                             13 : Ftue ue m fqef
                             12 : Estd td l epde
85
           Caesar
           Caesar
                             11 : Drsc sc k docd
86
           Caesar
                             10 : Cqrb rb j cnbc
87
                              9 : Bpqa qa i bmab
           Caesar
88
           Caesar
                              8 : Aopz pz h alza
89
                              7 : Znoy oy g zkyz
90
           Caesar
                             6 : Ymnx nx f yjxy
           Caesar
91
           Caesar
                             5 : Xlmw mw e xiwx
92
           Caesar
                              4 : Wklv lv d whvw
93
           Caesar
                              3 : Vjku ku c vguv
94
                              2 : Uijt jt b uftu
           Caesar
95
                              1 : This is a test
           Caesar
96
97
           Caesar
                              0 : Sghr hr z sdrs
98
99
    Original String: Hi my name is Robbie
    Encrypted String: Gh lx mzld hr Qnaahd
100
    Decrpyted String: Hi my name is Robbie
101
102
    Solve:
           Caesar 26: Gh lx mzld hr Qnaahd
103
           Caesar
                             25 : Fg kw lykc gq Pmzzgc
104
                             24 : Ef jv kxjb fp Olyyfb
           Caesar
105
                             23 : De iu jwia eo Nkxxea
           Caesar
106
           Caesar
                             22 : Cd ht ivhz dn Mjwwdz
107
                             21 : Bc gs hugy cm Livvcy
           Caesar
108
109
           Caesar
                             20 : Ab fr gtfx bl Khuubx
           Caesar
                             19 : Za eq fsew ak Jgttaw
110
           Caesar
                             18 : Yz dp erdv zj Ifsszv
111
           Caesar
                             17 : Xy co dqcu yi Herryu
112
           Caesar
                             16 : Wx bn cpbt xh Gdqqxt
113
                             15 : Vw am boas wg Fcppws
114
           Caesar
           Caesar
                             14 : Uv zl anzr vf Eboovr
115
           Caesar
                             13 : Tu yk zmyq ue Dannuq
116
           Caesar
                             12 : St xj ylxp td Czmmtp
117
           Caesar
                             11 : Rs wi xkwo sc Byllso
118
                             10 : Qr vh wjvn rb Axkkrn
119
           Caesar
                              9 : Pq ug vium qa Zwjjqm
120
           Caesar
           Caesar
                              8 : Op tf uhtl pz Yviipl
121
                              7 : No se tgsk oy Xuhhok
           Caesar
122
           Caesar
                              6 : Mn rd sfrj nx Wtggnj
123
           Caesar
                              5 : Lm qc reqi mw Vsffmi
124
                              4 : Kl pb qdph lv Ureelh
           Caesar
125
126
           Caesar
                              3 : Jk oa pcog ku Tqddkg
           Caesar
                              2 : Ij nz obnf jt Spccjf
127
           Caesar
                              1 : Hi my name is Robbie
128
                              0 : Gh lx mzld hr Qnaahd
           Caesar
129
130
    Original String: WandaVision
131
    Encrypted String: VzmczUhrhnm
132
    Decrpyted String: WandaVision
133
   Solve:
134
           Caesar 26: VzmczUhrhnm
135
```

```
Caesar
                              25 : UylbyTgqgml
136
                              24 : TxkaxSfpflk
            Caesar
137
            Caesar
138
                              23 : SwjzwReoekj
            Caesar
                              22 : RviyvQdndji
139
            Caesar
                              21 : QuhxuPcmcih
140
                              20 : PtgwtOblbhg
            Caesar
141
            Caesar
                              19 : OsfvsNakagf
142
143
            Caesar
                              18 : NreurMzjzfe
                              17 : MqdtqLyiyed
            Caesar
144
                              16 : LpcspKxhxdc
145
            Caesar
            Caesar
                              15 : KobroJwgwcb
146
147
            Caesar
                              14 : JnaqnIvfvba
            Caesar
                              13 : ImzpmHueuaz
148
149
            Caesar
                              12 : HlyolGtdtzy
                              11 : GkxnkFscsyx
150
            Caesar
            Caesar
                              10 : FjwmjErbrxw
151
            Caesar
                               9 : EivliDqaqwv
152
            Caesar
                               8 : DhukhCpzpvu
153
            Caesar
                               7 : CgtjgBoyout
154
                               6 : BfsifAnxnts
155
            Caesar
                               5 : AerheZmwmsr
            Caesar
156
157
            Caesar
                               4 : ZdqgdYlvlrq
            Caesar
                               3 : YcpfcXkukqp
158
            Caesar
                               2 : XboebWjtjpo
159
            Caesar
                               1 : WandaVision
160
            Caesar
                               0 : VzmczUhrhnm
161
162
    Original String: Abed
163
164
    Encrypted String: Zadc
    Decrpyted String: Abed
165
166
167
            Caesar 26: Zadc
            Caesar
                              25 : Yzcb
168
                              24 : Xyba
169
            Caesar
170
            Caesar
                              23 : Wxaz
            Caesar
                              22 : Vwzy
171
            Caesar
                              21 : Uvyx
172
                              20 : Tuxw
            Caesar
173
174
            Caesar
                              19 : Stwv
            Caesar
                              18 : Rsvu
175
            Caesar
                              17 : Qrut
176
            Caesar
                              16 : Pqts
177
            Caesar
                              15 : Opsr
178
                              14 : Norq
179
            Caesar
            Caesar
                              13 : Mnqp
180
            Caesar
                              12 : Lmpo
181
            Caesar
                              11 : Klon
182
            Caesar
                              10 : Jknm
183
                               9 : Ijml
            Caesar
184
                              8 : Hilk
185
            Caesar
                              7 : Ghkj
186
            Caesar
                               6 : Fgji
            Caesar
187
            Caesar
                               5 : Efih
188
                               4 : Dehg
            Caesar
189
            Caesar
                               3 : Cdgf
190
191
            Caesar
                               2 : Bcfe
            Caesar
                               1 : Abed
192
            Caesar
                               0 : Zadc
193
194
195
    Original String: The Mandalorian
    Encrypted String: Sgd Lzmczknqhzm
196
    Decrpyted String: The Mandalorian
197
198
    Solve:
            Caesar 26: Sgd Lzmczknqhzm
199
                              25 : Rfc Kylbyjmpgyl
200
```

```
Caesar
                             24 : Qeb Jxkaxilofxk
201
                             23 : Pda Iwjzwhknewj
           Caesar
202
203
           Caesar
                             22 : Ocz Hviyvgjmdvi
           Caesar
                             21 : Nby Guhxufilcuh
204
           Caesar
                             20 : Max Ftgwtehkbtg
205
                             19 : Lzw Esfvsdgjasf
           Caesar
206
                             18 : Kyv Dreurcfizre
207
           Caesar
208
           Caesar
                             17 : Jxu Cqdtqbehyqd
                             16 : Iwt Bpcspadgxpc
           Caesar
209
210
           Caesar
                             15 : Hvs Aobrozcfwob
                             14 : Gur Znaqnybevna
           Caesar
211
           Caesar
                             13 : Ftq Ymzpmxadumz
212
           Caesar
                             12 : Esp Xlyolwzctly
213
214
           Caesar
                             11 : Dro Wkxnkvybskx
                             10 : Cqn Vjwmjuxarjw
           Caesar
215
                              9 : Bpm Uivlitwzqiv
           Caesar
216
           Caesar
                              8 : Aol Thukhsvyphu
217
           Caesar
                              7 : Znk Sgtjgruxogt
218
                              6 : Ymj Rfsifqtwnfs
           Caesar
219
                              5 : Xli Qerhepsvmer
220
           Caesar
                              4 : Wkh Pdqgdoruldq
           Caesar
221
           Caesar
                              3 : Vjg Ocpfcnqtkcp
222
                              2 : Uif Nboebmpsjbo
           Caesar
223
           Caesar
                              1 : The Mandalorian
224
                              0 : Sgd Lzmczknqhzm
225
           Caesar
226
    Original String: Wow I learned Fortran I think
227
    Encrypted String: Vnv H kdzqmdc Enqsqzm H sghmj
228
229
    Decryyted String: Wow I learned Fortran I think
    Solve:
230
           Caesar 26: Vnv H kdzqmdc Enqsqzm H sghmj
231
232
           Caesar
                              25 : Umu G jcyplcb Dmprpyl G rfgli
           Caesar
                              24 : Tlt F ibxokba Cloqoxk F qefkh
233
           Caesar
                              23 : Sks E hawnjaz Bknpnwj E pdejg
234
           Caesar
                             22 : Rjr D gzvmizy Ajmomvi D ocdif
235
                             21 : Qiq C fyulhyx Zilnluh C nbche
           Caesar
236
           Caesar
                             20 : Php B extkgxw Yhkmktg B mabgd
237
                             19 : Ogo A dwsjfwv Xgjljsf A lzafc
           Caesar
238
239
           Caesar
                             18 : Nfn Z cvrievu Wfikire Z kyzeb
           Caesar
                             17 : Mem Y bughdut Vehjhqd Y jxyda
240
           Caesar
                             16 : Ldl X atpgcts Udgigpc X iwxcz
241
                             15 : Kck W zsofbsr Tcfhfob W hvwby
           Caesar
242
                             14 : Jbj V yrnearq Sbegena V guvax
           Caesar
243
                             13 : Iai U xqmdzqp Radfdmz U ftuzw
244
           Caesar
           Caesar
                             12 : Hzh T wplcypo Qzcecly T estyv
245
           Caesar
                             11 : Gyg S vokbxon Pybdbkx S drsxu
246
                             10 : Fxf R unjawnm Oxacajw R cqrwt
           Caesar
247
           Caesar
                              9 : Ewe Q tmizvml Nwzbziv Q bpqvs
248
                              8 : Dvd P slhyulk Mvyayhu P aopur
249
           Caesar
                              7 : Cuc O rkgxtkj Luxzxgt O znotq
250
           Caesar
           Caesar
                              6 : Btb N qjfwsji Ktwywfs N ymnsp
251
           Caesar
                              5 : Asa M pievrih Jsvxver M xlmro
252
           Caesar
                              4 : Zrz L ohduqhg Iruwudq L wklqn
253
           Caesar
                              3 : Yqy K ngctpgf Hqtvtcp K vjkpm
254
                              2 : Xpx J mfbsofe Gpsusbo J uijol
           Caesar
255
256
           Caesar
                              1 : Wow I learned Fortran I think
           Caesar
                              0 : Vnv H kdzqmdc Enqsqzm H sghmj
257
```

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.

```
$ SET SOURCEFORMAT "FREE"
  IDENTIFICATION DIVISION.
  PROGRAM-ID. IDEONE.
  AUTHOR. Robert Perrone.
  DATE-WRITTEN. March 22nd 2021
  *>ENVIRONMENT DIVISION.
  DATA DIVISION.
9
  WORKING-STORAGE SECTION.
10
  *> Original String Table
11
12 01 testStrTable.*> PIC X(30).
          02 testStr1 PIC X(30) VALUE "IBM".
          02 testStr2 PIC X(30) VALUE "Hello World".
14
```

```
02 testStr3 PIC X(30) VALUE "This is a test".
15
           02 testStr4 PIC X(30) VALUE "Hi my name is Robbie".
16
           02 testStr5 PIC X(30) VALUE "WandaVision".
17
           02 testStr6 PIC X(30) VALUE "Abed".
18
           02 testStr7 PIC X(30) VALUE "The Mandalorian".
19
           02 testStr8 PIC X(30) VALUE "Wow I learned COBOL I think".
20
  01 FILLER REDEFINES testStrTable.
21
           02 testStrTb OCCURS 8 INDEXED BY I.
22
                   03 testStrTb OCCURS 1 TIMES.
23
24
                            04 testStr PIC X(30).
25
  *> Encrypted String Table
26
  01 eTestStrTable PIC X(30).
27
28
           02 eTestStr1 PIC X(30) VALUE "".
           02 eTestStr2 PIC X(30) VALUE "".
29
           02 eTestStr3 PIC X(30) VALUE "".
30
           02 eTestStr4 PIC X(30) VALUE "".
31
           02 eTestStr5 PIC X(30) VALUE "".
32
           02 eTestStr6 PIC X(30) VALUE "".
33
           02 eTestStr7 PIC X(30) VALUE "".
34
           02 eTestStr8 PIC X(30) VALUE "".
35
  01 FILLER REDEFINES eTestStrTable.
           02 eTestStrTb OCCURS 8 TIMES INDEXED BY J.
37
                   03 eTestStrTb OCCURS 1 TIMES.
38
                            04 eTestStr PIC X(30).
39
40
41 01 shiftAmount PIC S9(4) VALUE -1.
42 01 maxShiftAmount PIC 9(2) VALUE 26.
44 01 tempStr PIC X(30) VALUE "".
45 01 encryptedStr PIC X(30) VALUE "".
46 01 decryptedStr PIC X(30) VALUE "".
  01 solvedStr PIC X(30) VALUE "".
47
  *>01 I PIC 9(1) VALUE 1.
49
  *>01 J PIC 9(1) VALUE 1.
50
51
  PROCEDURE DIVISION.
52
53
           your code goes here
54
           DISPLAY "Caesar Cipher ".
55
           DISPLAY "".
56
57
           SET I TO 1.
58
           SET J TO 1.
59
           begin.
                    PERFORM iForLoop VARYING I FROM 1 BY 1 UNTIL I>8.
61
                   STOP RUN.
62
63
64
           iForLoop.
                    {\tt MOVE \ Function \ UPPER-CASE(testStr(I)) \ TO \ tempStr.}
65
                    DISPLAY "Original String: " tempStr.
66
67
                    {\tt CALL \ "ENCRYPT" \ USING \ tempStr, \ shiftAmount, \ encryptedStr.}
68
                    DISPLAY "Encrypted String: " encryptedStr.
69
70
                    {\tt CALL \ "DECRYPT" \ USING \ encryptedStr.}
71
                   DISPLAY "Decrypted String: " decryptedStr.
72
73
74
                    DISPLAY "Solve:".
                    CALL "SOLVE" USING encryptedStr, maxShiftAmount, solvedStr.
75
                    DISPLAY "".
76
                    *{\tt >MOVE} \ \ Function \ \ solve(encryptedTempStr, maxShiftAmount) \ \ TO \ \ decryptedTempStr.
77
  END PROGRAM IDEONE.
78
79
```

```
80 *> Encrypt Subprogram
81 IDENTIFICATION DIVISION.
82 PROGRAM - ID. ENCRYPT.
83 DATA DIVISION.
84 WORKING-STORAGE SECTION.
85 01 letter PIC X(1).
86 01 newLetter PIC X(1).
87 01 asciiVal PIC 9(2).
88 01 newAsciiVal PIC 9(2).
90 01 E PIC 9(2) VALUE 1.
91 01 len PIC 9(3) VALUE 0.
92
93 LINKAGE SECTION.
94 01 tempStr PIC X(30).
95 01 shiftAmount PIC S9(4).
   01 encryptedStr PIC X(30).
97
   PROCEDURE DIVISION USING tempStr, shiftAmount, encryptedStr.
98
           *>MOVE "returned" TO encryptedStr.
99
           *>DISPLAY "ENCRYPT".
100
           begin.
101
                    SET shiftAmount TO -1.
102
                    MOVE Function LENGTH(tempStr) TO len.
103
                    *>DISPLAY len.
104
                    PERFORM eForLoop VARYING E FROM 1 BY 1 UNTIL E > len.
105
                             EXIT PROGRAM.
106
            eForLoop.
107
108
                    MOVE tempStr(E:E) TO letter.
                    *>DISPLAY letter.
109
                    MOVE Function ORD(letter) TO asciiVal.
110
                    *>COMPUTE asciiVal = asciiVal - 1.
111
                    *>DISPLAY asciiVal.
112
113
                    *> If character from A to Z
114
                    IF (asciiVal >= 66) AND (asciiVal <= 91) THEN
115
                             *>If character is A and shift is negative, wrap around
116
                             IF (asciiVal = 66) AND (shiftAmount < 0) THEN
117
118
                                     COMPUTE newAsciiVal = asciiVal + 26 + shiftAmount.
                                     MOVE Function CHAR(newAsciiVal) TO newLetter.
119
                                     MOVE newLetter TO encryptedStr(E:E).
120
121
                             *>If character is Z and shift is positive, wrap around
122
                             IF ((asciiVal = 91) AND (shiftAmount > 0)) THEN
123
                                     COMPUTE newAsciiVal = asciiVal - 26 + shiftAmount.
124
                                     MOVE Function CHAR(newAsciiVal) TO newLetter.
125
                                     MOVE newLetter TO encryptedStr(E:E).
126
                             *>ELSE
127
                             IF NOT ((asciiVal = 66) AND (shiftAmount < 0)) AND NOT ((asciiVal = 91) AND (shiftAmount < 0))
128
                                     *>DISPLAY letter.
129
                                     *>DISPLAY asciiVal.
130
                                     COMPUTE newAsciiVal = asciiVal + shiftAmount.
131
                                      *>DISPLAY shiftAmount.
132
                                      *>DISPLAY newAsciiVal.
133
                                     MOVE Function CHAR(newAsciiVal) TO newLetter.
134
135
                                     MOVE newLetter TO encryptedStr(E:E).
                                     *>DISPLAY encryptedStr(E:E).
136
137
                    IF NOT ((asciiVal >= 66) AND (asciiVal <= 91)) THEN
138
139
                             *>If character is space or other
                             MOVE letter TO encryptedStr(E:E).
140
141
142 END PROGRAM ENCRYPT.
143
144 *>Decrypt Subprogram
```

```
145 IDENTIFICATION DIVISION.
146 PROGRAM - ID. DECRYPT.
147 DATA DIVISION.
148 WORKING-STORAGE SECTION.
149 01 letter PIC X(1).
150 01 newLetter PIC X(1).
151 01 asciiVal PIC 9(2).
152 01 newAsciiVal PIC 9(2).
153
154 01 D PIC 9(2) VALUE 1.
155 01 len PIC 9(3) VALUE 0.
156
157 LINKAGE SECTION.
158 01 encryptedStr PIC X(30).
   01 shiftAmount PIC S9(4).
   01 decryptedStr PIC X(30).
160
161
   {\tt PROCEDURE\ DIVISION\ USING\ encryptedStr,\ shiftAmount,\ decryptedStr.}
162
           *>MOVE "returned" TO encryptedStr.
163
           *>DISPLAY "DECRYPT".
164
           begin.
165
                    MOVE Function LENGTH(encryptedStr) TO len.
166
                    *>DISPLAY len.
167
                    MULTIPLY shiftAmount BY -1 GIVING shiftAmount.
168
                    PERFORM dForLoop VARYING D FROM 1 BY 1 UNTIL D > len.
169
                             EXIT PROGRAM.
170
171
            dForLoop.
                    MOVE encryptedStr(D:D) TO letter.
172
173
                    *>DISPLAY letter.
                    MOVE Function ORD(letter) TO asciiVal.
174
                    *>COMPUTE asciiVal = asciiVal - 1.
175
176
                    *>DISPLAY asciiVal.
                    *> If character from A to Z
177
178
                    *>Make shift amount opposite sign
179
180
181
                    IF (asciiVal) >= 66 AND (asciiVal <= 91) THEN
182
183
                             *>If character is A and shift is negative, wrap around
                             IF (asciiVal = 66) AND (shiftAmount < 0) THEN</pre>
184
                                      COMPUTE newAsciiVal = asciiVal + 26 + shiftAmount.
185
                                      MOVE Function CHAR(newAsciiVal) TO newLetter.
186
                                      MOVE newLetter TO decryptedStr(D:D).
187
188
                             *>If character is Z and shift is positive, wrap around
189
                             IF (asciiVal = 91) AND (shiftAmount > 0) THEN
190
                                      COMPUTE newAsciiVal = asciiVal - 26 + shiftAmount.
191
                                      MOVE Function CHAR(newAsciiVal) TO newLetter.
192
                                      MOVE newLetter TO decryptedStr(D:D).
193
                             *>ELSE
194
                             IF NOT ((asciiVal = 66) AND (shiftAmount < 0)) AND NOT ((asciiVal = 91) AND (shiftAmount)
195
                                      *>DISPLAY letter.
196
                                      *>DISPLAY asciiVal.
197
                                      COMPUTE newAsciiVal = asciiVal + shiftAmount.
198
                                      *>DISPLAY shiftAmount.
199
200
                                      *>DISPLAY newAsciiVal.
                                      MOVE Function CHAR (newAsciiVal) TO newLetter.
201
                                      MOVE newLetter TO decryptedStr(D:D).
202
                                      *>DISPLAY encryptedStr(D:D).
203
204
                    IF NOT ((asciiVal >= 66) AND (asciiVal <= 91)) THEN
205
                             *>If character is space or other
206
                             MOVE letter TO decryptedStr(D:D).
207
208
209 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.
227 LINKAGE SECTION.
228 01 encryptedStr PIC X(30).
229 01 maxShiftAmount PIC 9(2).
   01 solvedStr PIC X(30).
230
231
   PROCEDURE DIVISION USING encryptedStr, maxShiftAmount, solvedStr. 
 *>MOVE "returned" TO solvedStr.
232
233
            *>DISPLAY "SOLVE".
234
            begin.
235
                    MOVE Function LENGTH(solvedStr) TO len.
236
                    SET shiftAmount TO -1.
237
238
                    MOVE encryptedStr TO solvedStr.
                    DISPLAY "Caesar 26: " solvedStr.
239
                    PERFORM jForLoop VARYING J FROM 1 BY 1 UNTIL J > maxShiftAmount.
240
                             EXIT PROGRAM.
241
242
            jForLoop.
243
                    PERFORM sForLoop VARYING S FROM 1 BY 1 UNTIL S > len.
244
                    COMPUTE caesarNum = maxShiftAmount - j.
245
                    DISPLAY "Caesar " caesarNum ": " solvedStr.
246
247
248
            sForLoop.
                    MOVE solvedStr(S:S) TO letter.
249
                    *>DISPLAY letter.
250
                    MOVE Function ORD(letter) TO asciiVal.
251
                    *>COMPUTE asciiVal = asciiVal - 1.
252
                    *>DISPLAY asciiVal.
253
                    *> If character from A to Z
254
                    IF (asciiVal >= 66) AND (asciiVal <= 91) THEN
                             *>If character is A and shift is negative, wrap around
256
                             IF (asciiVal = 66) AND (shiftAmount < 0) THEN
257
                                      COMPUTE newAsciiVal = asciiVal + 26 + shiftAmount.
258
                                      MOVE Function CHAR(newAsciiVal) TO newLetter.
259
                                      MOVE newLetter TO solvedStr(S:S).
260
                             *>If character is Z and shift is positive, wrap around
261
                             IF (asciiVal = 91) AND (shiftAmount > 0) THEN
262
                                      COMPUTE newAsciiVal = asciiVal - 26 + shiftAmount.
263
                                      MOVE Function CHAR(newAsciiVal) TO newLetter.
264
265
                                      MOVE newLetter TO solvedStr(S:S).
                             *>ELSE
266
                             IF NOT ((asciiVal = 66) AND (shiftAmount < 0)) AND NOT ((asciiVal = 91) AND (shiftAmount
267
                                      *>DISPLAY letter.
268
269
                                      *>DISPLAY asciiVal.
                                      COMPUTE newAsciiVal = asciiVal + shiftAmount.
270
                                      *>DISPLAY shiftAmount.
271
272
                                      *>DISPLAY newAsciiVal.
                                      MOVE Function CHAR (newAsciiVal) TO newLetter.
273
                                      MOVE newLetter TO solvedStr(S:S).
274
```

```
IF NOT ((asciiVal >= 66) AND (asciiVal <= 91)) THEN

*>If character is space or other

MOVE letter TO solvedStr(S:S).

END PROGRAM SOLVE.
```

```
Caesar Cipher
3 Original String: IBM
4 Encrypted String: HAL
5 Decrypted String: IBM
6 Solve:
  Caesar 26: HAL
8 Caesar 25: GZ
9 Caesar 24: FY
10 Caesar 23: EX
11 Caesar 22: DW
12 Caesar 21: CV
13 Caesar 20: BU
14 Caesar 19: AT
15 Caesar 18: ZS
16 Caesar 17: YR
17 Caesar 16: XQ
18 Caesar 15: WP
19 Caesar 14: VO
20 Caesar 13: UN
21 Caesar 12: TM
22 Caesar 11: SL
23 Caesar 10: RK
24 Caesar 09: QJ
25 Caesar 08: PI
26 Caesar 07: OH
27 Caesar 06: NG
28 Caesar 05: MF
29 Caesar 04: LE
30 Caesar 03: KD
31 Caesar 02: JC
32 Caesar 01: IB
33 Caesar 00: HA
_{35} Original String: HELLO WORLD
36 Encrypted String: GDKKN VNQKC
37 Decrypted String: HELLO WORLD
38 Solve:
39 Caesar 26: GDKKN VNQKC
40 Caesar 25: FC
41 Caesar 24: EB
42 Caesar 23: DA
43 Caesar 22: CZ
44 Caesar 21: BY
45 Caesar 20: AX
46 Caesar 19: ZW
47 Caesar 18: YV
48 Caesar 17: XU
49 Caesar 16: WT
50 Caesar 15: VS
51 Caesar 14: UR
52 Caesar 13: TQ
53 Caesar 12: SP
54 Caesar 11: RO
55 Caesar 10: QN
```

```
56 Caesar 09: PM
57 Caesar 08: OL
58 Caesar 07: NK
59 Caesar 06: MJ
60 Caesar 05: LI
61 Caesar 04: KH
62 Caesar 03: JG
63 Caesar 02: IF
64 Caesar 01: HE
65 Caesar 00: GD
67 Original String: THIS IS A TEST
68 Encrypted String: SGHR HR Z SDRS
69 Decrypted String: THIS IS A TEST
70 Solve:
71 Caesar 26: SGHR HR Z SDRS
72 Caesar 25: RF
73 Caesar 24: QE
74 Caesar 23: PD
75 Caesar 22: OC
76 Caesar 21: NB
77 Caesar 20: MA
78 Caesar 19: LZ
79 Caesar 18: KY
80 Caesar 17: JX
81 Caesar 16: IW
82 Caesar 15: HV
83 Caesar 14: GU
84 Caesar 13: FT
85 Caesar 12: ES
86 Caesar 11: DR
87 Caesar 10: CQ
88 Caesar 09: BP
89 Caesar 08: AO
90 Caesar 07: ZN
91 Caesar 06: YM
92 Caesar 05: XL
93 Caesar 04: WK
94 Caesar 03: VJ
95 Caesar 02: UI
96 Caesar 01: TH
97 Caesar 00: SG
98
99 Original String: HI MY NAME IS ROBBIE
100 Encrypted String: GH LX MZLD HR QNAAHD
101 Decrypted String: HI MY NAME IS ROBBIE
102 Solve:
103 Caesar 26: GH LX MZLD HR QNAAHD
104 Caesar 25: FG
105 Caesar 24: EF
106 Caesar 23: DE
107 Caesar 22: CD
108 Caesar 21: BC
109 Caesar 20: AB
110 Caesar 19: ZA
111 Caesar 18: YZ
112 Caesar 17: XY
113 Caesar 16: WX
114 Caesar 15: VW
115 Caesar 14: UV
116 Caesar 13: TU
117 Caesar 12: ST
118 Caesar 11: RS
119 Caesar 10: QR
120 Caesar 09: PQ
```

```
121 Caesar 08: OP
122 Caesar 07: NO
123 Caesar 06: MN
124 Caesar 05: LM
125 Caesar 04: KL
126 Caesar 03: JK
127 Caesar 02: IJ
128 Caesar 01: HI
129 Caesar 00: GH
130
131 Original String: WANDAVISION
132 Encrypted String: VZMCZUHRHNM
133 Decrypted String: WANDAVISION
134 Solve:
135 Caesar 26: VZMCZUHRHNM
136 Caesar 25: UY
137 Caesar 24: TX
138 Caesar 23: SW
139 Caesar 22: RV
140 Caesar 21: QU
141 Caesar 20: PT
142 Caesar 19: OS
143 Caesar 18: NR
144 Caesar 17: MQ
145 Caesar 16: LP
146 Caesar 15: KO
147 Caesar 14: JN
148 Caesar 13: IM
149 Caesar 12: HL
150 Caesar 11: GK
151 Caesar 10: FJ
152 Caesar 09: EI
153 Caesar 08: DH
154 Caesar 07: CG
155 Caesar 06: BF
156 Caesar 05: AE
157 Caesar 04: ZD
158 Caesar 03: YC
159 Caesar 02: XB
160 Caesar 01: WA
161 Caesar 00: VZ
162
163 Original String: ABED
164 Encrypted String: ZADC
165 Decrypted String: ABED
166 Solve:
167 Caesar 26: ZADC
168 Caesar 25: YZ
169 Caesar 24: XY
170 Caesar 23: WX
171 | Caesar 22: VW
172 Caesar 21: UV
173 Caesar 20: TU
174 Caesar 19: ST
175 Caesar 18: RS
176 Caesar 17: QR
177 Caesar 16: PQ
178 Caesar 15: OP
179 Caesar 14: NO
180 Caesar 13: MN
181 | Caesar 12: LM
182 Caesar 11: KL
183 Caesar 10: JK
184 Caesar 09: IJ
185 Caesar 08: HI
```

```
186 Caesar 07: GH
187 Caesar 06: FG
188 Caesar 05: EF
189 Caesar 04: DE
190 Caesar 03: CD
191 Caesar 02: BC
192 Caesar 01: AB
193 Caesar 00: ZA
194
195 Original String: THE MANDALORIAN
196 Encrypted String: SGD LZMCZKNQHZM
197 Decrypted String: THE MANDALORIAN
198 Solve:
199 Caesar 26: SGD LZMCZKNQHZM
200 Caesar 25: RF
201 Caesar 24: QE
202 Caesar 23: PD
203 Caesar 22: OC
204 Caesar 21: NB
205 Caesar 20: MA
206 Caesar 19: LZ
207 Caesar 18: KY
208 Caesar 17: JX
209 Caesar 16: IW
210 Caesar 15: HV
211 Caesar 14: GU
212 Caesar 13: FT
213 Caesar 12: ES
214 Caesar 11: DR
215 Caesar 10: CQ
216 Caesar 09: BP
217 Caesar 08: AO
218 Caesar 07: ZN
219 Caesar 06: YM
220 Caesar 05: XL
221 Caesar 04: WK
222 Caesar 03: VJ
223 Caesar 02: UI
224 Caesar 01: TH
225 Caesar 00: SG
227 Original String: WOW I LEARNED COBOL I THINK
228 Encrypted String: VNV H KDZQMDC BNANK H SGHMJ
229 Decrypted String: WOW I LEARNED COBOL I THINK
230 Solve:
231 Caesar 26: VNV H KDZQMDC BNANK H SGHMJ
232 Caesar 25: UM
233 Caesar 24: TL
234 Caesar 23: SK
235 Caesar 22: RJ
236 Caesar 21: QI
237 Caesar 20: PH
238 Caesar 19: OG
239 Caesar 18: NF
240 Caesar 17: ME
241 Caesar 16: LD
242 Caesar 15: KC
243 Caesar 14: JB
244 Caesar 13: IA
245 Caesar 12: HZ
246 Caesar 11: GY
247 Caesar 10: FX
248 Caesar 09: EW
249 Caesar 08: DV
250 Caesar 07: CU
```

```
251 Caesar 06: BT
252 Caesar 05: AS
253 Caesar 04: ZR
254 Caesar 02: XP
255 Caesar 01: WO
256 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.

```
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
10
                                             encryptedStr += Chr(Asc(letter) + 26 + shift)
11
                                    else if((Asc(letter) = 90) and (shift > 0)) then
12
                                             encryptedStr += Chr(Asc(letter) - 26 + shift)
13
14
                                    else
                                             encryptedStr += Chr(Asc(letter) + shift)
15
16
                                    End if
                           Else
17
                                    encryptedStr += letter
18
                           End if
19
                   Next
20
```

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

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

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

```
37 Decrypted String: HELLO WORLD
   Solve:
38
           Caesar 26: GDKKN VNQKC
39
           Caesar 25: FCJJM UMPJB
40
           Caesar 24: EBIIL TLOIA
41
           Caesar 23: DAHHK SKNHZ
42
           Caesar 22: CZGGJ RJMGY
43
           Caesar 21: BYFFI QILFX
44
           Caesar 20: AXEEH PHKEW
45
           Caesar 19: ZWDDG OGJDV
46
           Caesar 18: YVCCF NFICU
47
48
           Caesar 17: XUBBE MEHBT
           Caesar 16: WTAAD LDGAS
49
50
           Caesar 15: VSZZC KCFZR
           Caesar 14: URYYB JBEYQ
51
           Caesar 13: TQXXA IADXP
52
           Caesar 12: SPWWZ HZCWO
           Caesar 11: ROVVY GYBVN
54
           Caesar 10: QNUUX FXAUM
55
           Caesar 9: PMTTW EWZTL
56
           Caesar 8: OLSSV DVYSK
57
           Caesar 7: NKRRU CUXRJ
           Caesar 6: MJQQT BTWQI
59
           Caesar 5: LIPPS ASVPH
60
           Caesar 4: KHOOR ZRUOG
61
           Caesar 3: JGNNQ YQTNF
62
           Caesar 2: IFMMP XPSME
           Caesar 1: HELLO WORLD
64
           Caesar O: GDKKN VNQKC
67 Original String: THIS IS A TEST
68 Encrypted String: SGHR HR Z SDRS
  Decrypted String: THIS IS A TEST
69
70
  Solve:
71
           Caesar 26: SGHR HR Z SDRS
           Caesar 25: RFGQ GQ Y RCQR
72
           Caesar 24: QEFP FP X QBPQ
73
           Caesar 23: PDEO EO W PAOP
74
           Caesar 22: OCDN DN V OZNO
75
           Caesar 21: NBCM CM U NYMN
76
           Caesar 20: MABL BL T MXLM
77
           Caesar 19: LZAK AK S LWKL
78
           Caesar 18: KYZJ ZJ R KVJK
79
           Caesar 17: JXYI YI Q JUIJ
80
           Caesar 16: IWXH XH P ITHI
81
           Caesar 15: HVWG WG O HSGH
           Caesar 14: GUVF VF N GRFG
83
           Caesar 13: FTUE UE M FQEF
84
           Caesar 12: ESTD TD L EPDE
85
           Caesar 11: DRSC SC K DOCD
86
           Caesar 10: CQRB RB J CNBC
           Caesar 9: BPQA QA I BMAB
88
           Caesar 8: AOPZ PZ H ALZA
89
           Caesar 7: ZNOY OY G ZKYZ
90
           Caesar 6: YMNX NX F YJXY
91
92
           Caesar 5: XLMW MW E XIWX
           Caesar 4: WKLV LV D WHVW
93
           Caesar 3: VJKU KU C VGUV
94
           Caesar 2: UIJT JT B UFTU
95
           Caesar 1: THIS IS A TEST
           Caesar O: SGHR HR Z SDRS
97
98
99 Original String: HI MY NAME IS ROBBIE
100 Encrypted String: GH LX MZLD HR QNAAHD
101 Decrypted String: HI MY NAME IS ROBBIE
```

102 Solve: Caesar 26: GH LX MZLD HR QNAAHD 103 104 Caesar 25: FG KW LYKC GQ PMZZGC Caesar 24: EF JV KXJB FP OLYYFB 105 Caesar 23: DE IU JWIA EO NKXXEA 106 Caesar 22: CD HT IVHZ DN MJWWDZ 107 Caesar 21: BC GS HUGY CM LIVVCY 108 Caesar 20: AB FR GTFX BL KHUUBX 109 Caesar 19: ZA EQ FSEW AK JGTTAW 110 Caesar 18: YZ DP ERDV ZJ IFSSZV 111 Caesar 17: XY CO DQCU YI HERRYU 112 Caesar 16: WX BN CPBT XH GDQQXT 113 Caesar 15: VW AM BOAS WG FCPPWS 114 115 Caesar 14: UV ZL ANZR VF EBOOVR Caesar 13: TU YK ZMYQ UE DANNUQ 116 Caesar 12: ST XJ YLXP TD CZMMTP 117 Caesar 11: RS WI XKWO SC BYLLSO 118 Caesar 10: QR VH WJVN RB AXKKRN 119 Caesar 9: PQ UG VIUM QA ZWJJQM 120 Caesar 8: OP TF UHTL PZ YVIIPL 121 Caesar 7: NO SE TGSK OY XUHHOK 122 Caesar 6: MN RD SFRJ NX WTGGNJ Caesar 5: LM QC REQI MW VSFFMI 124 Caesar 4: KL PB QDPH LV UREELH 125 Caesar 3: JK OA PCOG KU TQDDKG 126 Caesar 2: IJ NZ OBNF JT SPCCJF 127 Caesar 1: HI MY NAME IS ROBBIE 128 Caesar O: GH LX MZLD HR QNAAHD 129 130 Original String: WANDAVISION 131 Encrypted String: VZMCZUHRHNM 132 133 Decrypted String: WANDAVISION Solve: 134 Caesar 26: VZMCZUHRHNM 135 Caesar 25: UYLBYTGQGML 136 Caesar 24: TXKAXSFPFLK 137 Caesar 23: SWJZWREOEKJ 138 Caesar 22: RVIYVQDNDJI 139 140 Caesar 21: QUHXUPCMCIH Caesar 20: PTGWTOBLBHG 141 Caesar 19: OSFVSNAKAGF 142 Caesar 18: NREURMZJZFE 143 Caesar 17: MQDTQLYIYED 144 Caesar 16: LPCSPKXHXDC 145 Caesar 15: KOBROJWGWCB 146 Caesar 14: JNAQNIVFVBA 147 Caesar 13: IMZPMHUEUAZ 148 Caesar 12: HLYOLGTDTZY 149 Caesar 11: GKXNKFSCSYX 150 Caesar 10: FJWMJERBRXW 151 Caesar 9: EIVLIDQAQWV Caesar 8: DHUKHCPZPVU 153 Caesar 7: CGTJGBOYOUT 154 Caesar 6: BFSIFANXNTS 155 Caesar 5: AERHEZMWMSR 156 157 Caesar 4: ZDQGDYLVLRQ Caesar 3: YCPFCXKUKQP 158 Caesar 2: XBOEBWJTJPO 159 Caesar 1: WANDAVISION 160 Caesar O: VZMCZUHRHNM 161 162 163 Original String: ABED 164 Encrypted String: ZADC 165 Decrypted String: ABED 166 Solve:

```
Caesar 26: ZADC
167
           Caesar 25: YZCB
168
169
           Caesar 24: XYBA
           Caesar 23: WXAZ
170
           Caesar 22: VWZY
171
           Caesar 21: UVYX
172
           Caesar 20: TUXW
173
174
           Caesar 19: STWV
           Caesar 18: RSVU
175
           Caesar 17: QRUT
176
           Caesar 16: PQTS
177
           Caesar 15: OPSR
178
           Caesar 14: NORQ
179
180
           Caesar 13: MNQP
           Caesar 12: LMPO
181
           Caesar 11: KLON
182
           Caesar 10: JKNM
183
           Caesar 9: IJML
184
           Caesar 8: HILK
185
           Caesar 7: GHKJ
186
           Caesar 6: FGJI
187
           Caesar 5: EFIH
           Caesar 4: DEHG
189
            Caesar 3: CDGF
190
           Caesar 2: BCFE
191
           Caesar 1: ABED
192
           Caesar 0: ZADC
193
194
   Original String: THE MANDALORIAN
   Encrypted String: SGD LZMCZKNQHZM
196
   Decrypted String: THE MANDALORIAN
197
198
   Solve:
           Caesar 26: SGD LZMCZKNQHZM
199
           Caesar 25: RFC KYLBYJMPGYL
200
           Caesar 24: QEB JXKAXILOFXK
201
           Caesar 23: PDA IWJZWHKNEWJ
202
           Caesar 22: OCZ HVIYVGJMDVI
203
           Caesar 21: NBY GUHXUFILCUH
204
205
           Caesar 20: MAX FTGWTEHKBTG
           Caesar 19: LZW ESFVSDGJASF
206
           Caesar 18: KYV DREURCFIZRE
207
           Caesar 17: JXU CQDTQBEHYQD
208
           Caesar 16: IWT BPCSPADGXPC
209
           Caesar 15: HVS AOBROZCFWOB
210
           Caesar 14: GUR ZNAQNYBEVNA
211
           Caesar 13: FTQ YMZPMXADUMZ
           Caesar 12: ESP XLYOLWZCTLY
213
           Caesar 11: DRO WKXNKVYBSKX
214
           Caesar 10: CQN VJWMJUXARJW
215
           Caesar 9: BPM UIVLITWZQIV
216
217
           Caesar 8: AOL THUKHSVYPHU
           Caesar 7: ZNK SGTJGRUXOGT
218
           Caesar 6: YMJ RFSIFQTWNFS
219
           Caesar 5: XLI QERHEPSVMER
220
           Caesar 4: WKH PDQGDORULDQ
221
222
           Caesar 3: VJG OCPFCNQTKCP
           Caesar 2: UIF NBOEBMPSJBO
223
224
           Caesar 1: THE MANDALORIAN
           Caesar O: SGD LZMCZKNQHZM
225
227 Original String: WOW I LEARNED BASIC I THINK
   Encrypted String: VNV H KDZQMDC AZRHB H SGHMJ
228
  Decrypted String: WOW I LEARNED BASIC I THINK
229
230 Solve:
            Caesar 26: VNV H KDZQMDC AZRHB H SGHMJ
231
```

```
Caesar 25: UMU G JCYPLCB ZYQGA G RFGLI
232
           Caesar 24: TLT F IBXOKBA YXPFZ F QEFKH
233
234
           Caesar 23: SKS E HAWNJAZ XWOEY E PDEJG
           Caesar 22: RJR D GZVMIZY WVNDX D
235
           Caesar 21: QIQ C FYULHYX VUMCW C NBCHE
           Caesar 20: PHP B EXTKGXW UTLBV B MABGD
237
           Caesar 19: OGO A DWSJFWV TSKAU A LZAFC
238
239
           Caesar 18: NFN Z CVRIEVU SRJZT Z KYZEB
           Caesar 17: MEM Y BUQHDUT RQIYS
240
241
           Caesar 16: LDL X ATPGCTS QPHXR X IWXCZ
           Caesar 15: KCK W ZSOFBSR POGWQ W HVWBY
242
           Caesar 14: JBJ V YRNEARQ ONFVP V GUVAX
243
           Caesar 13: IAI U XQMDZQP NMEUO U FTUZW
244
           Caesar 12: HZH T WPLCYPO MLDTN
245
           Caesar 11: GYG S VOKBXON LKCSM S DRSXU
           Caesar 10: FXF R UNJAWNM KJBRL R CQRWT
247
           Caesar 9: EWE Q TMIZVML JIAQK Q BPQVS
248
           Caesar 8: DVD P SLHYULK IHZPJ P AOPUR
249
           Caesar 7: CUC O RKGXTKJ HGYOI O ZNOTQ
250
           Caesar 6: BTB N QJFWSJI GFXNH N YMNSP
251
           Caesar 5: ASA M PIEVRIH FEWMG M XLMRO
252
           Caesar 4: ZRZ L OHDUQHG EDVLF L WKLQN
           Caesar 3: YQY K NGCTPGF DCUKE K VJKPM
254
           Caesar 2:
                     XPX J MFBSOFE CBTJD
                                          J UIJOL
255
           Caesar 1: WOW I LEARNED BASIC I THINK
256
           Caesar O: VNV H KDZQMDC AZRHB H SGHMJ
257
```

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.

```
1 program ideone;
  (*define encrypt function*)
  function encrypt(testStr: string; shift: integer): string;
4
           encryptedChrArr : array of char;
          letter : char;
6
7
           encryptedStr : string;
          x, e : integer;
  BEGIN
9
      encryptedStr := '';
10
           SetLength(encryptedChrArr, length(testStr));
11
12
          for x:=0 to length(testStr)-1 do
13
          BEGIN
14
                   encryptedChrArr[x] := testStr[x+1]; (*Add each character of string to a char array*)
                   (*writeln(encryptedChrArr[x]);*)
16
          END; {for x}
17
18
          for e:=0 to length(encryptedChrArr)-1 do
19
          BEGIN
20
                   letter := encryptedChrArr[e];
21
                   (*writeln(letter);*) (*check letters*)
22
                   if((Integer(letter) >= 65)  and (Integer(letter) <= 90))  then BEGIN(*A to Z*)
23
                            if((Integer(letter) = 65) and (shift < 0)) then BEGIN(*If A and shift is negative,
24
25
                                    encryptedChrArr[e] := Chr(Integer(letter) + 26 + shift);
                       END else if((Integer(letter) = 90) and (shift > 0)) then BEGIN(*If Z, and shift is posi
26
27
                                    encryptedChrArr[e] := Chr(Integer(letter) - 26 + shift);
                       END else BEGIN(*Perform a normal shift*)
28
                                    encryptedChrArr[e] := Chr(Integer(letter) + shift);
                            END:
30
                   END else BEGIN (*Other Characters*)
31
32
                            encryptedChrArr[e] := letter;
                   END; {if}
33
                   (*turn character array into new string*)
                   encryptedStr := encryptedStr + encryptedChrArr[e];
35
36
          END; {for e}
37
           encrypt := encryptedStr;
  END; {function encrypt}
38
39
  (*define decrypt function*)
40
  function decrypt(testStr: string; shift: integer): string;
41
42
43
          decryptedChrArr : array of char;
          letter : char;
44
          decryptedStr : string;
45
46
          y, d : integer;
47
  BEGIN
48
       decryptedStr := '';
49
           SetLength(decryptedChrArr, length(testStr));
50
51
          for y:=0 to length(testStr)-1 do
52
          BEGIN
53
                   decryptedChrArr[y] := testStr[y+1]; (*Add each character of string to a char array*)
54
                   (*writeln(decryptedChrArr[x]);*)
55
56
          END; {for y}
57
          for d:=0 to length(decryptedChrArr)-1 do
          BEGIN
59
```

```
letter := decryptedChrArr[d];
60
                    (*writeln(letter);*) (*check letters*)
61
                    if((Integer(letter) >= 65) and (Integer(letter) <= 90)) then BEGIN(*A to Z*)
62
                            if((Integer(letter) = 65) and (-shift < 0)) then BEGIN(*If A and shift is negative;
63
                                     decryptedChrArr[d] := Chr(Integer(letter) + 26 - shift);
64
                        END else if((Integer(letter) = 90) and (-shift > 0)) then BEGIN(*If Z, and shift is pos
65
                                     decryptedChrArr[d] := Chr(Integer(letter) - 26 - shift);
66
67
                        END else BEGIN(*Perform a normal shift*)
                                     decryptedChrArr[d] := Chr(Integer(letter) - shift);
68
69
                            END:
                    END else BEGIN (*Other Characters*)
70
71
                            decryptedChrArr[d] := letter;
                    END; {if}
72
73
                    (*turn character array into new string*)
                    decryptedStr := decryptedStr + decryptedChrArr[d];
74
           END: {for d}
75
           decrypt := decryptedStr;
  END; {function decrpyt}
77
78
  (*define solve function*)
80
81 function solve(testStr: string; maxShift: integer) : string;
82
           solvedChrArr : array of char;
83
           letter : char;
84
           solvedStr : string;
85
           shift : integer;
           tempNum : integer;
87
88
           n, j, s : integer;
89
  BEGIN
90
           writeln(#9'Caesar 26: ' + testStr);
91
           solvedStr := '';
92
           shift := -1;
93
           SetLength(solvedChrArr, length(testStr));
94
95
           for n:=0 to length(testStr)-1 do
96
97
98
                    solvedChrArr[n] := testStr[n+1]; (*Add each character of string to a char array*)
                    (*writeln(decryptedChrArr[x]);*)
99
           END; {for n}
100
101
           for j:=0 to maxShift-1 do
102
103
           BEGIN
                    solvedStr := '';
104
                    for s:=0 to length(solvedChrArr)-1 do
105
                    BEGIN
106
107
                            letter := solvedChrArr[s];
                             (*writeln(letter);*) (*check letters*)
108
                            if((Integer(letter) >= 65) and (Integer(letter) <= 90)) then BEGIN(*A to Z*)
109
                                     if ((Integer (letter) = 65) and (shift < 0)) then BEGIN (*If A and shift is no
110
                                             solvedChrArr[s] := Chr(Integer(letter) + 26 + shift);
111
                                 END else if((Integer(letter) = 90) and (shift > 0)) then BEGIN(*If Z, and shift
112
                                              solvedChrArr[s] := Chr(Integer(letter) - 26 + shift);
113
                                 END else BEGIN(*Perform a normal shift*)
114
115
                                             solvedChrArr[s] := Chr(Integer(letter) + shift);
                                     END:
116
                            END else BEGIN (*Other Characters*)
117
                                     solvedChrArr[s] := letter;
118
119
120
                             (*turn character array into new string*)
                            solvedStr := solvedStr + solvedChrArr[s];
121
                    END; {for s}
122
                    tempNum := maxShift-j-1;
123
                    writeln(#9'Caesar ', tempNum, ': ' + solvedStr);
124
```

```
END; {for j}
125
           solve := solvedStr;
126
127
   END; {function solve}
128
   var
129
           testStr1, testStr2, testStr3, testStr4, testStr5, testStr6, testStr7, testStr8 : stripg;
130
           shiftAmount, maxShiftAmount : integer;
131
132
           testStrArr : array[0..7] of string;
133
134
           tempStr, encryptedTempStr, decryptedTempStr, solvedTempStr : string;
           i : integer;
135
136
   BEGIN
137
138
           writeln('Caesar Cipher');
           writeln();
139
           testStr1 := 'IBM';
140
           testStr2 := 'Hello World';
141
           testStr3 := 'This is a test';
142
           testStr4 := 'Hi my name is Robbie';
143
           testStr5 := 'WandaVision';
144
           testStr6 := 'Abed';
145
           testStr7 := 'The Mandalorian';
           testStr8 := 'Wow I learned Pascal I think';
147
148
149
           shiftAmount := -1;
           maxShiftAmount := 26;
150
151
           testStrArr[0] := testStr1; testStrArr[1] := testStr2;
152
153
           testStrArr[2] := testStr3; testStrArr[3] := testStr4;
           testStrArr[4] := testStr5; testStrArr[5] := testStr6;
154
           testStrArr[6] := testStr7; testStrArr[7] := testStr8;
155
156
           for i:=0 to length(testStrArr)-1 do
157
           BEGIN
158
                    tempStr := UpCase(testStrArr[i]); (*Change all letters to uppercase*)
159
                    writeln('Original String: ' + tempStr);
160
161
                    encryptedTempStr := encrypt(tempStr, shiftAmount);
162
163
                    writeln('Encrypted String: ' + encryptedTempStr);
164
                    decryptedTempStr := decrypt(encryptedTempStr, shiftAmount);
165
                    writeln('Decrypted String: ' + decryptedTempStr);
166
167
168
                    writeln('Solve: ');
                    solvedTempStr := solve(encryptedTempStr, maxShiftAmount);
169
                    writeln();
170
           END; {for i}
171
   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
13
           Caesar 19: ATE
14
           Caesar 18: ZSD
           Caesar 17: YRC
16
           Caesar 16: XQB
17
           Caesar 15: WPA
18
           Caesar 14: VOZ
19
           Caesar 13: UNY
20
           Caesar 12: TMX
21
           Caesar 11: SLW
22
           Caesar 10: RKV
23
24
           Caesar 9: QJU
           Caesar 8: PIT
25
26
           Caesar 7: OHS
           Caesar 6: NGR
27
           Caesar 5: MFQ
28
           Caesar 4: LEP
           Caesar 3: KDO
30
           Caesar 2: JCN
31
           Caesar 1: IBM
32
           Caesar 0: HAL
33
35 Original String: HELLO WORLD
  Encrypted String: GDKKN VNQKC
36
  Decrypted String: HELLO WORLD
37
  Solve:
38
           Caesar 26: GDKKN VNQKC
39
           Caesar 25: FCJJM UMPJB
40
           Caesar 24: EBIIL TLOIA
41
           Caesar 23: DAHHK SKNHZ
42
           Caesar 22: CZGGJ RJMGY
43
           Caesar 21: BYFFI QILFX
44
           Caesar 20: AXEEH PHKEW
45
           Caesar 19: ZWDDG OGJDV
46
           Caesar 18: YVCCF NFICU
47
           Caesar 17: XUBBE MEHBT
48
           Caesar 16: WTAAD LDGAS
49
           Caesar 15: VSZZC KCFZR
50
           Caesar 14: URYYB JBEYQ
51
           Caesar 13: TQXXA IADXP
52
           Caesar 12: SPWWZ HZCWO
53
           Caesar 11: ROVVY GYBVN
54
           Caesar 10: QNUUX FXAUM
55
           Caesar 9: PMTTW EWZTL
56
           Caesar 8: OLSSV DVYSK
57
           Caesar 7: NKRRU CUXRJ
           Caesar 6: MJQQT BTWQI
59
           Caesar 5: LIPPS ASVPH
60
           Caesar 4: KHOOR ZRUOG
61
           Caesar 3: JGNNQ YQTNF
62
63
           Caesar 2: IFMMP XPSME
           Caesar 1: HELLO WORLD
64
           Caesar O: GDKKN VNQKC
65
66
  Original String: THIS IS A TEST
67
  Encrypted String: SGHR HR Z SDRS
  Decrypted String: THIS IS A TEST
69
70
  Solve:
           Caesar 26: SGHR HR Z SDRS
71
           Caesar 25: RFGQ GQ Y RCQR
           Caesar 24: QEFP FP X QBPQ
73
           Caesar 23: PDEO EO W PAOP
74
           Caesar 22: OCDN DN V OZNO
75
           Caesar 21: NBCM CM U NYMN
76
           Caesar 20: MABL BL T MXLM
77
```

```
Caesar 19: LZAK AK S LWKL
           Caesar 18: KYZJ ZJ R KVJK
79
           Caesar 17: JXYI YI Q JUIJ
           Caesar 16: IWXH XH P ITHI
81
           Caesar 15: HVWG WG O HSGH
82
           Caesar 14: GUVF VF N GRFG
83
           Caesar 13: FTUE UE M FQEF
84
           Caesar 12: ESTD TD L EPDE
85
           Caesar 11: DRSC SC K DOCD
86
           Caesar 10: CQRB RB J CNBC
87
           Caesar 9: BPQA QA I BMAB
88
           Caesar 8: AOPZ PZ H ALZA
89
           Caesar 7: ZNOY OY G ZKYZ
90
91
           Caesar 6: YMNX NX F YJXY
           Caesar 5: XLMW MW E XIWX
92
           Caesar 4: WKLV LV D WHVW
93
           Caesar 3: VJKU KU C VGUV
           Caesar 2: UIJT JT B UFTU
95
           Caesar 1: THIS IS A TEST
96
           Caesar O: SGHR HR Z SDRS
97
98
   Original String: HI MY NAME IS ROBBIE
   Encrypted String: GH LX MZLD HR QNAAHD
100
   Decrypted String: HI MY NAME IS ROBBIE
101
102
   Solve:
           Caesar 26: GH LX MZLD HR QNAAHD
103
           Caesar 25: FG KW LYKC GQ PMZZGC
104
           Caesar 24: EF JV KXJB FP OLYYFB
105
106
           Caesar 23: DE IU JWIA EO NKXXEA
           Caesar 22: CD HT IVHZ DN MJWWDZ
107
           Caesar 21: BC GS HUGY CM LIVVCY
108
           Caesar 20: AB FR GTFX BL KHUUBX
109
           Caesar 19: ZA EQ FSEW AK JGTTAW
110
           Caesar 18: YZ DP ERDV ZJ IFSSZV
111
           Caesar 17: XY CO DQCU YI HERRYU
112
           Caesar 16: WX BN CPBT XH GDQQXT
113
           Caesar 15: VW AM BOAS WG FCPPWS
114
           Caesar 14: UV ZL ANZR VF EBOOVR
115
116
           Caesar 13: TU YK ZMYQ UE DANNUQ
           Caesar 12: ST XJ YLXP TD CZMMTP
117
           Caesar 11: RS WI XKWO SC BYLLSO
118
           Caesar 10: QR VH WJVN RB AXKKRN
119
           Caesar 9: PQ UG VIUM QA ZWJJQM
120
           Caesar 8: OP TF UHTL PZ YVIIPL
121
           Caesar 7: NO SE TGSK OY XUHHOK
122
           Caesar 6: MN RD SFRJ NX WTGGNJ
123
           Caesar 5: LM QC REQI MW VSFFMI
124
           Caesar 4: KL PB QDPH LV UREELH
125
           Caesar 3: JK OA PCOG KU TQDDKG
126
           Caesar 2: IJ NZ OBNF JT SPCCJF
127
           Caesar 1: HI MY NAME IS ROBBIE
128
           Caesar O: GH LX MZLD HR QNAAHD
129
130
   Original String: WANDAVISION
131
   Encrypted String: VZMCZUHRHNM
132
133
   Decrypted String: WANDAVISION
   Solve:
134
           Caesar 26: VZMCZUHRHNM
135
           Caesar 25: UYLBYTGQGML
136
137
           Caesar 24: TXKAXSFPFLK
           Caesar 23: SWJZWREOEKJ
138
           Caesar 22: RVIYVQDNDJI
139
           Caesar 21: QUHXUPCMCIH
140
           Caesar 20: PTGWTOBLBHG
141
           Caesar 19: OSFVSNAKAGF
142
```

```
Caesar 18: NREURMZJZFE
143
            Caesar 17: MQDTQLYIYED
144
145
            Caesar 16: LPCSPKXHXDC
            Caesar 15: KOBROJWGWCB
146
            Caesar 14: JNAQNIVFVBA
147
           Caesar 13: IMZPMHUEUAZ
148
            Caesar 12: HLYOLGTDTZY
149
            Caesar 11: GKXNKFSCSYX
150
            Caesar 10: FJWMJERBRXW
151
           Caesar 9: EIVLIDQAQWV
152
           Caesar 8: DHUKHCPZPVU
153
            Caesar 7: CGTJGBOYOUT
154
            Caesar 6: BFSIFANXNTS
155
156
            Caesar 5: AERHEZMWMSR
            Caesar 4: ZDQGDYLVLRQ
157
            Caesar 3: YCPFCXKUKQP
158
            Caesar 2: XBOEBWJTJPO
159
            Caesar 1: WANDAVISION
160
            Caesar O: VZMCZUHRHNM
161
162
   Original String: ABED
163
   Encrypted String: ZADC
   Decrypted String: ABED
165
   Solve:
166
            Caesar 26: ZADC
167
            Caesar 25: YZCB
168
           Caesar 24: XYBA
169
            Caesar 23: WXAZ
170
            Caesar 22: VWZY
171
            Caesar 21: UVYX
172
            Caesar 20: TUXW
173
174
           Caesar 19: STWV
            Caesar 18: RSVU
175
            Caesar 17: QRUT
176
            Caesar 16: PQTS
177
            Caesar 15: OPSR
178
            Caesar 14: NORQ
179
            Caesar 13: MNQP
180
           Caesar 12: LMPO
181
           Caesar 11: KLON
182
            Caesar 10: JKNM
183
            Caesar 9: IJML
184
            Caesar 8: HILK
185
            Caesar 7: GHKJ
186
            Caesar 6: FGJI
187
            Caesar 5: EFIH
188
            Caesar 4: DEHG
189
            Caesar 3: CDGF
190
            Caesar 2: BCFE
191
            Caesar 1: ABED
192
            Caesar 0: ZADC
194
   Original String: THE MANDALORIAN
195
   Encrypted String: SGD LZMCZKNQHZM
196
   Decrypted String: THE MANDALORIAN
197
198
   Solve:
            Caesar 26: SGD LZMCZKNQHZM
199
200
            Caesar 25: RFC KYLBYJMPGYL
            Caesar 24: QEB JXKAXILOFXK
201
202
            Caesar 23: PDA IWJZWHKNEWJ
            Caesar 22: OCZ HVIYVGJMDVI
203
            Caesar 21: NBY GUHXUFILCUH
204
            Caesar 20: MAX FTGWTEHKBTG
205
            Caesar 19: LZW ESFVSDGJASF
206
```

Caesar 18: KYV DREURCFIZRE

207

```
Caesar 17: JXU CQDTQBEHYQD
208
           Caesar 16: IWT BPCSPADGXPC
209
210
           Caesar 15: HVS AOBROZCFWOB
           Caesar 14: GUR ZNAQNYBEVNA
211
           Caesar 13: FTQ YMZPMXADUMZ
212
           Caesar 12: ESP XLYOLWZCTLY
213
           Caesar 11: DRO WKXNKVYBSKX
214
           Caesar 10: CQN VJWMJUXARJW
215
           Caesar 9: BPM UIVLITWZQIV
216
217
           Caesar 8: AOL THUKHSVYPHU
           Caesar 7: ZNK SGTJGRUXOGT
218
           Caesar 6: YMJ RFSIFQTWNFS
219
           Caesar 5: XLI QERHEPSVMER
220
221
           Caesar 4: WKH PDQGDORULDQ
           Caesar 3: VJG OCPFCNQTKCP
222
           Caesar 2: UIF NBOEBMPSJBO
223
           Caesar 1: THE MANDALORIAN
           Caesar O: SGD LZMCZKNQHZM
225
226
  Original String: WOW I LEARNED PASCAL I THINK
227
   Encrypted String: VNV H KDZQMDC OZRBZK H SGHMJ
228
   Decrypted String: WOW I LEARNED PASCAL I THINK
230
   Solve:
           Caesar 26: VNV H KDZQMDC OZRBZK H SGHMJ
231
           Caesar 25: UMU G JCYPLCB NYQAYJ G RFGLI
232
           Caesar 24: TLT F IBXOKBA MXPZXI F QEFKH
233
           Caesar 23: SKS E HAWNJAZ LWOYWH E PDEJG
234
           Caesar 22: RJR D GZVMIZY KVNXVG D OCDIF
235
236
           Caesar 21: QIQ C FYULHYX JUMWUF C NBCHE
           Caesar 20: PHP B EXTKGXW ITLVTE B MABGD
237
           Caesar 19: OGO A DWSJFWV HSKUSD A LZAFC
238
           Caesar 18: NFN Z CVRIEVU GRJTRC Z KYZEB
239
           Caesar 17: MEM Y BUQHDUT FQISQB Y JXYDA
240
           Caesar 16: LDL X ATPGCTS EPHRPA X IWXCZ
241
           Caesar 15: KCK W ZSOFBSR DOGQOZ W HVWBY
242
           Caesar 14: JBJ V YRNEARQ CNFPNY V GUVAX
243
           Caesar 13: IAI U XQMDZQP BMEOMX U FTUZW
244
           Caesar 12: HZH T WPLCYPO ALDNLW T ESTYV
245
246
           Caesar 11: GYG S VOKBXON ZKCMKV S DRSXU
           Caesar 10: FXF R UNJAWNM YJBLJU R CQRWT
247
           Caesar 9: EWE Q TMIZVML XIAKIT Q BPQVS
248
           Caesar 8: DVD P SLHYULK WHZJHS P AOPUR
249
           Caesar 7: CUC O RKGXTKJ VGYIGR O ZNOTQ
250
           Caesar 6: BTB N QJFWSJI UFXHFQ N YMNSP
251
           Caesar 5: ASA M PIEVRIH TEWGEP M XLMRO
252
           Caesar 4: ZRZ L OHDUQHG SDVFDO L WKLQN
           Caesar 3: YQY K NGCTPGF RCUECN K VJKPM
254
           Caesar 2: XPX J MFBSOFE QBTDBM J UIJOL
255
           Caesar 1: WOW I LEARNED PASCAL I THINK
256
           Caesar O: VNV H KDZQMDC OZRBZK H SGHMJ
257
```

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.

```
object Main {
       def main(args: Array[String]) {
2
           println("Caesar Cipher\n")
           val testStr1 = "IBM"
           val testStr2 = "Hello World"
           val testStr3 = "This is a test"
6
           val testStr4 = "Hi my name is Robbie"
           val testStr5 = "WandaVision"
           val testStr6 = "Abed"
           val testStr7 = "The Mandalorian"
10
           val testStr8 = "Wow I learned Scala I think"
11
12
           val shiftAmount = -1
13
               val maxShiftAmount = 26
14
15
               var testStrArr = Array(testStr1, testStr2, testStr3, testStr4, testStr5, testStr6, testStr7, testStr7, testStr8
16
17
               for(i <- testStrArr) {</pre>
18
                   var tempStr = i.toUpperCase()
19
                   println("Original String: " + tempStr)
20
21
                   var encryptedTempStr = encrypt(tempStr, shiftAmount)
22
                   println("Encrypted String: " + encryptedTempStr)
23
24
                   var decryptedTempStr = decrypt(encryptedTempStr, shiftAmount)
25
                   println("Decrypted String: " + decryptedTempStr)
26
27
                   println("Solve: ")
28
                   solve(encryptedTempStr, maxShiftAmount)
29
```

```
println()
30
               }
31
32
      def encrypt(testStr:String, shift:Int) : String /*Array[Char]*/= {
33
           //var encryptedStr = "
34
           var encryptedChrArr = testStr.toCharArray()
35
           var e = 0
36
37
           for(e <- 0 until testStr.length()) {</pre>
38
39
               var letter = encryptedChrArr(e) //testStr.charAt(e)
               //println(letter)
40
               if(letter.toInt >= 65 && letter.toInt <= 90) { //A to Z
41
                   if(letter.toInt == 65 && shift < 0) { //If A and shift is negative, then loop around to Z
42
43
                        encryptedChrArr(e) = (letter.toInt + 26 + shift).toChar
                   } else if(letter.toInt == 90 && shift > 0) { //If Z, and shift is positive, then loop around
44
                        encryptedChrArr(e) = (letter.toInt - 26 + shift).toChar
45
                   } else { // Else perform a normal shift
46
                        encryptedChrArr(e) = (letter.toInt + shift).toChar
47
                        //encryptedStr.concat(letter.toString)
48
49
                        //encryptedChrArr(e) = (testStr.charAt(e).toInt + shift)
                        //println(letter)
50
                   }
51
52
               }
               else { //Other Characters
53
54
                   //Other Characters should not be changed in Caesar Cipher
                   encryptedChrArr(e) = letter
55
56
                   //println(letter)
               }
57
58
           }
           var encryptedStr = encryptedChrArr.mkString("")
59
60
           return encryptedStr
61
      def decrypt(testStr:String, shift:Int) : String = {
62
           var decryptedChrArr = testStr.toCharArray()
63
           var d = 0
64
65
           for(d <- 0 until testStr.length()) {</pre>
66
               var letter = decryptedChrArr(d)
67
               //shift = -shift//Switch sign of shift
68
               //println(letter)
69
               if(letter.toInt >= 65 && letter.toInt <= 90) { //A to Z
70
                   if(letter.toInt == 65 && -shift < 0) { //If A and shift is negative, then loop around to Z
71
                        decryptedChrArr(d) = (letter.toInt + 26 - shift).toChar
72
                   } else if(letter.toInt == 90 && -shift > 0) { //If Z, and shift is positive, then loop around
73
                        decryptedChrArr(d) = (letter.toInt - 26 - shift).toChar
74
                   } else { // Else perform a normal shift
75
                        decryptedChrArr(d) = (letter.toInt - shift).toChar
76
77
                        //println(letter)
78
79
               else { //Other Characters
                   //Other Characters should not be changed in Caesar Cipher
81
                   decryptedChrArr(d) = letter
82
                   //println(letter)
83
84
85
           }
           var decryptedStr = decryptedChrArr.mkString("")
86
           return decryptedStr
87
88
      def solve(testStr:String, maxShift:Int) {
89
           var solvedChrArr = testStr.toCharArray()
90
           val shift = -1
91
           var j = 0
92
           var s = 0
93
94
```

```
println("\tCaesar " + (26).toString + ": " + testStr)
95
           for(j <- 0 until maxShift) {</pre>
96
97
                    for(s <- 0 until testStr.length()) {</pre>
                        var letter = solvedChrArr(s)
98
                         //println(letter)
99
                        if(letter.toInt >= 65 && letter.toInt <= 90) { //A to Z
100
                             if(letter.toInt == 65 && shift < 0) { //If A and shift is negative, then loop arour
101
102
                                 solvedChrArr(s) = (letter.toInt + 26 + shift).toChar
                             } else if(letter.toInt == 90 && shift > 0) { //If Z, and shift is positive, then lo
103
104
                                 solvedChrArr(s) = (letter.toInt - 26 + shift).toChar
                             } else { // Else perform a normal shift
105
                                 solvedChrArr(s) = (letter.toInt + shift).toChar
106
                                 //println(letter)
107
108
                        }
109
                        else { //Other Characters
110
                             solvedChrArr(s) = letter
111
                             //println(letter)
112
113
                    }
114
                    var solvedStr = solvedChrArr.mkString("")
115
                    println("\tCaesar" + (maxShift-j-1).toString + ":" + solvedStr)
116
           }
117
       }
118
119
   }
```

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

```
_{36}| Encrypted String: GDKKN VNQKC
   Decrypted String: HELLO WORLD
38
   Solve:
            Caesar 26: GDKKN VNQKC
39
           Caesar 25: FCJJM UMPJB
40
           Caesar 24: EBIIL TLOIA
41
           Caesar 23: DAHHK SKNHZ
42
           Caesar 22: CZGGJ RJMGY
43
           Caesar 21: BYFFI QILFX
44
           Caesar 20: AXEEH PHKEW
45
           Caesar 19: ZWDDG OGJDV
46
47
           Caesar 18: YVCCF NFICU
           Caesar 17: XUBBE MEHBT
48
49
           Caesar 16: WTAAD LDGAS
           Caesar 15: VSZZC KCFZR
50
           Caesar 14: URYYB JBEYQ
51
           Caesar 13: TQXXA IADXP
52
           Caesar 12: SPWWZ HZCWO
53
           Caesar 11: ROVVY GYBVN
54
           Caesar 10: QNUUX FXAUM
55
           Caesar 9: PMTTW EWZTL
56
           Caesar 8: OLSSV DVYSK
           Caesar 7: NKRRU CUXRJ
58
           Caesar 6: MJQQT BTWQI
59
           Caesar 5: LIPPS ASVPH
60
           Caesar 4: KHOOR ZRUOG
61
           Caesar 3: JGNNQ YQTNF
           Caesar 2: IFMMP XPSME
63
           Caesar 1: HELLO WORLD
           Caesar O: GDKKN VNQKC
65
67 Original String: THIS IS A TEST
   Encrypted String: SGHR HR Z SDRS Decrypted String: THIS IS A TEST
68
70
   Solve:
            Caesar 26: SGHR HR Z SDRS
71
           Caesar 25: RFGQ GQ Y RCQR
72
           Caesar 24: QEFP FP X QBPQ
73
           Caesar 23: PDEO EO W PAOP
74
           Caesar 22: OCDN DN V OZNO
75
           Caesar 21: NBCM CM U NYMN
76
           Caesar 20: MABL BL T MXLM
77
           Caesar 19: LZAK AK S LWKL
78
           Caesar 18: KYZJ ZJ R KVJK
79
           Caesar 17: JXYI YI Q JUIJ
80
           Caesar 16: IWXH XH P ITHI
81
           Caesar 15: HVWG WG O HSGH
82
           Caesar 14: GUVF VF N GRFG
83
           Caesar 13: FTUE UE M FQEF
84
           Caesar 12: ESTD TD L EPDE
85
           Caesar 11: DRSC SC K DOCD
           Caesar 10: CQRB RB J CNBC
87
           Caesar 9: BPQA QA I BMAB
88
           Caesar 8: AOPZ PZ H ALZA
89
           Caesar 7: ZNOY OY G ZKYZ
90
91
           Caesar 6: YMNX NX F YJXY
           Caesar 5: XLMW MW E XIWX
92
           Caesar 4: WKLV LV D WHVW
           Caesar 3: VJKU KU C VGUV
94
           Caesar 2: UIJT JT B UFTU
           Caesar 1: THIS IS A TEST
96
           Caesar O: SGHR HR Z SDRS
97
99 Original String: HI MY NAME IS ROBBIE
100 Encrypted String: GH LX MZLD HR QNAAHD
```

101 Decrypted String: HI MY NAME IS ROBBIE Solve: 102 103 Caesar 26: GH LX MZLD HR QNAAHD Caesar 25: FG KW LYKC GQ PMZZGC 104 Caesar 24: EF JV KXJB FP OLYYFB 105 Caesar 23: DE IU JWIA EO NKXXEA 106 Caesar 22: CD HT IVHZ DN MJWWDZ 107 Caesar 21: BC GS HUGY CM LIVVCY 108 Caesar 20: AB FR GTFX BL KHUUBX 109 Caesar 19: ZA EQ FSEW AK JGTTAW 110 Caesar 18: YZ DP ERDV ZJ IFSSZV 111 Caesar 17: XY CO DQCU YI HERRYU 112 Caesar 16: WX BN CPBT XH GDQQXT 113 114 Caesar 15: VW AM BOAS WG FCPPWS Caesar 14: UV ZL ANZR VF EBOOVR 115 Caesar 13: TU YK ZMYQ UE DANNUQ 116 Caesar 12: ST XJ YLXP TD CZMMTP 117 Caesar 11: RS WI XKWO SC BYLLSO 118 Caesar 10: QR VH WJVN RB AXKKRN 119 Caesar 9: PQ UG VIUM QA ZWJJQM 120 Caesar 8: OP TF UHTL PZ YVIIPL 121 Caesar 7: NO SE TGSK OY XUHHOK Caesar 6: MN RD SFRJ NX WTGGNJ 123 Caesar 5: LM QC REQI MW VSFFMI 124 Caesar 4: KL PB QDPH LV UREELH 125 Caesar 3: JK OA PCOG KU TQDDKG 126 Caesar 2: IJ NZ OBNF JT SPCCJF 127 Caesar 1: HI MY NAME IS ROBBIE 128 129 Caesar O: GH LX MZLD HR QNAAHD 130 Original String: WANDAVISION 131 132 Encrypted String: VZMCZUHRHNM Decrypted String: WANDAVISION 133 Solve: 134 Caesar 26: VZMCZUHRHNM 135 Caesar 25: UYLBYTGQGML 136 Caesar 24: TXKAXSFPFLK 137 Caesar 23: SWJZWREOEKJ 138 Caesar 22: RVIYVQDNDJI 139 Caesar 21: QUHXUPCMCIH 140 Caesar 20: PTGWTOBLBHG 141 Caesar 19: OSFVSNAKAGF 142 Caesar 18: NREURMZJZFE 143 Caesar 17: MQDTQLYIYED 144 Caesar 16: LPCSPKXHXDC 145 Caesar 15: KOBROJWGWCB 146 Caesar 14: JNAQNIVFVBA 147 Caesar 13: IMZPMHUEUAZ 148 Caesar 12: HLYOLGTDTZY 149 Caesar 11: GKXNKFSCSYX 150 Caesar 10: FJWMJERBRXW Caesar 9: EIVLIDQAQWV 152 Caesar 8: DHUKHCPZPVU 153 Caesar 7: CGTJGBOYOUT 154 Caesar 6: BFSIFANXNTS 155 156 Caesar 5: AERHEZMWMSR Caesar 4: ZDQGDYLVLRQ 157 Caesar 3: YCPFCXKUKQP 158 Caesar 2: XBOEBWJTJPO 159 Caesar 1: WANDAVISION 160 Caesar O: VZMCZUHRHNM 161 162 163 Original String: ABED 164 Encrypted String: ZADC 165 Decrypted String: ABED

```
166 Solve:
           Caesar 26: ZADC
167
168
           Caesar 25: YZCB
           Caesar 24: XYBA
169
           Caesar 23: WXAZ
170
           Caesar 22: VWZY
171
           Caesar 21: UVYX
172
           Caesar 20: TUXW
173
           Caesar 19: STWV
174
           Caesar 18: RSVU
175
           Caesar 17: QRUT
176
           Caesar 16: PQTS
177
           Caesar 15: OPSR
178
179
           Caesar 14: NORQ
           Caesar 13: MNQP
180
           Caesar 12: LMPO
181
           Caesar 11: KLON
182
           Caesar 10: JKNM
183
           Caesar 9: IJML
184
           Caesar 8: HILK
185
           Caesar 7: GHKJ
186
           Caesar 6: FGJI
           Caesar 5: EFIH
188
            Caesar 4: DEHG
189
           Caesar 3: CDGF
190
           Caesar 2: BCFE
191
192
           Caesar 1: ABED
           Caesar 0: ZADC
193
194
   Original String: THE MANDALORIAN
195
   Encrypted String: SGD LZMCZKNQHZM
196
197
   Decrypted String: THE MANDALORIAN
   Solve:
198
            Caesar 26: SGD LZMCZKNQHZM
199
           Caesar 25: RFC KYLBYJMPGYL
200
           Caesar 24: QEB JXKAXILOFXK
201
           Caesar 23: PDA IWJZWHKNEWJ
202
           Caesar 22: OCZ HVIYVGJMDVI
203
204
           Caesar 21: NBY GUHXUFILCUH
           Caesar 20: MAX FTGWTEHKBTG
205
           Caesar 19: LZW ESFVSDGJASF
206
           Caesar 18: KYV DREURCFIZRE
207
           Caesar 17: JXU CQDTQBEHYQD
208
           Caesar 16: IWT BPCSPADGXPC
209
           Caesar 15: HVS AOBROZCFWOB
210
           Caesar 14: GUR ZNAQNYBEVNA
211
           Caesar 13: FTQ YMZPMXADUMZ
212
           Caesar 12: ESP XLYOLWZCTLY
213
           Caesar 11: DRO WKXNKVYBSKX
214
           Caesar 10: CQN VJWMJUXARJW
215
           Caesar 9: BPM UIVLITWZQIV
           Caesar 8: AOL THUKHSVYPHU
217
           Caesar 7: ZNK SGTJGRUXOGT
218
           Caesar 6: YMJ RFSIFQTWNFS
219
           Caesar 5: XLI QERHEPSVMER
220
221
           Caesar 4: WKH PDQGDORULDQ
           Caesar 3: VJG OCPFCNQTKCP
222
           Caesar 2: UIF NBOEBMPSJBO
223
           Caesar 1: THE MANDALORIAN
224
           Caesar O: SGD LZMCZKNQHZM
226
227 Original String: WOW I LEARNED SCALA I THINK
228 Encrypted String: VNV H KDZQMDC RBZKZ H SGHMJ
229 Decrypted String: WOW I LEARNED SCALA I THINK
230 Solve:
```

```
Caesar 26: VNV H KDZQMDC RBZKZ H SGHMJ
231
           Caesar 25: UMU G JCYPLCB QAYJY G RFGLI
232
           Caesar 24: TLT F IBXOKBA PZXIX F QEFKH
233
           Caesar 23: SKS E HAWNJAZ OYWHW E PDEJG
234
235
           Caesar 22: RJR D GZVMIZY NXVGV D OCDIF
           Caesar 21: QIQ C FYULHYX MWUFU C NBCHE
236
           Caesar 20: PHP B EXTKGXW LVTET B MABGD
237
           Caesar 19: OGO A DWSJFWV KUSDS A LZAFC
238
           Caesar 18: NFN Z CVRIEVU JTRCR Z KYZEB
239
           Caesar 17: MEM Y BUQHDUT ISQBQ Y JXYDA
240
           Caesar 16: LDL X ATPGCTS HRPAP X IWXCZ
241
242
           Caesar 15: KCK W ZSOFBSR GQOZO W HVWBY
           Caesar 14: JBJ V YRNEARQ FPNYN V GUVAX
243
244
           Caesar 13: IAI U XQMDZQP EOMXM U FTUZW
           Caesar 12: HZH T WPLCYPO DNLWL T ESTYV
245
           Caesar 11: GYG S VOKBXON CMKVK S DRSXU
246
           Caesar 10: FXF R UNJAWNM BLJUJ R CQRWT
           Caesar 9: EWE Q TMIZVML AKITI Q BPQVS
248
           Caesar 8: DVD P SLHYULK ZJHSH P AOPUR
249
           Caesar 7: CUC O RKGXTKJ YIGRG O ZNOTQ
250
           Caesar 6: BTB N QJFWSJI XHFQF N YMNSP
251
           Caesar 5: ASA M PIEVRIH WGEPE M XLMRO
           Caesar 4: ZRZ L OHDUQHG VFDOD L WKLQN
253
           Caesar 3: YQY K NGCTPGF UECNC K VJKPM
254
           Caesar 2: XPX J MFBSOFE TDBMB J UIJOL
255
           Caesar 1: WOW I LEARNED SCALA I THINK
256
           Caesar O: VNV H KDZQMDC RBZKZ H SGHMJ
```

6 Language Rankings

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