```
algol,n<
begin
   comment

        GC7J6KQ

   Time buffer:     3397.02s = 56m 37.02s
   Time nonbuffer: 13755.50s = 3h 49m 15.50s

   No buffer GIER:

   Time classic:        12949.57
   Time turbo:          12120.39 6.4pct

   Buffer GIER:

   Time classic:         3397.18
   Time turbo:           3077.07 9.4pct
   ;
   comment PERM code taken from APL/360 ADVANCEDEX PERM function;
   procedure PERM(Z,a,b);
   value a,b;
   integer a,b;
   integer array Z;
   begin
      integer i,j,rem;
      rem:=b-1;
      for i:=1 step 1 until a do
      begin
         Z[a-i+1]:=1+rem mod i;
         rem:=rem:i
      end;
      for i:=a-1 step -1 until 1 do
      for j:=i+1 step 1 until a do
      if Z[i]≤Z[j] then Z[j]:=Z[j]+1
   end PERM;
   integer procedure ord(s);
   string s;
   begin
      integer c1,c2,c3;
      boolean b;
      b:=boolean s;
      c1 := integer (b∧40 63);
      c2 := integer ((b shift -6)∧40 63);
      c3 := integer ((b shift -12)∧40 63);
      ord := if c1=60 then c2+128 else c1
   end;
   integer procedure xor(a,b);
   value a,b;
   integer a,b;
   xor:=integer (-,((boolean a) = (boolean b)));
   integer procedure fetch char(A,offset);
   value offset;
   integer array A;
   integer offset;
   begin
      integer i,j;
      i:=integer (((boolean offset)shift -3)∧3 0 37 m);
      j:=(integer((boolean offset)∧37 0 3 m))×5;
      fetch char := integer (((boolean A[i])shift -j)
            ∧ 35 0 5 m)
   end fetch char;
   procedure WRITE CHAR(c);
   value c;
```

```
integer c;
begin
   if -,((c>127) = (case>127)) then
   begin
      case:=128-case;
      writechar(58+case:64)
   end change case;
   writechar(c mod 128)
end WRITECHAR;
integer array cipher[0:285];
integer array lookfor[0:2];
integer cipherlen,lookforlen;
integer array baudot[0:31];
integer case,iperm;
real procedure clock count;
code clock count;
1, 37;
  zl        , grf p-1   ; RF:=clock count; stack[p-1]:=RF;
e;
select(32);
clock count;
baudot[0]:=ord(<</>);
baudot[0]:=ord(<<2>);
baudot[1]:=ord(<<T>);
baudot[2]:=ord(<<3>);
baudot[3]:=ord(<<O>);
baudot[4]:=ord(<<9>);
baudot[5]:=ord(<<H>);
baudot[6]:=ord(<<N>);
baudot[7]:=ord(<<M>);
baudot[8]:=ord(<<4>);
baudot[9]:=ord(<<L>);
baudot[10]:=ord(<<R>);
baudot[11]:=ord(<<G>);
baudot[12]:=ord(<<I>);
baudot[13]:=ord(<<P>);
baudot[14]:=ord(<<C>);
baudot[15]:=ord(<<V>);
baudot[16]:=ord(<<E>);
baudot[17]:=ord(<<Z>);
baudot[18]:=ord(<<D>);
baudot[19]:=ord(<<B>);
baudot[20]:=ord(<<S>);
baudot[21]:=ord(<<Y>);
baudot[22]:=ord(<<F>);
baudot[23]:=ord(<<X>);
baudot[24]:=ord(<<A>);
baudot[25]:=ord(<<W>);
baudot[26]:=ord(<<J>);
baudot[27]:=ord(<<+>);
baudot[27]:=ord(<<5>);
baudot[28]:=ord(<<U>);
baudot[29]:=ord(<<Q>);
baudot[30]:=ord(<<K>);
baudot[31]:=ord(<<8>);
begin comment read baudot;
   integer i;
   integer array revbaudot[0:255];
   integer procedure LYN;
   begin
      integer c;
again:c:=lyn;
      if c=58 ∨ c=60 then
      begin
```

```
            case:=(c-58)×64;
            goto again
         end;
         LYN:=c+case
      end LYN;
      integer procedure read baudot(A);
      integer array A;
      begin
         integer len,c,i,j;
         len:=0;
again:   c:=LYN;
         if c=64 ∨ c=192 then goto out;
         i:=len:8;
         j:=(len mod 8)×5;
         c:=revbaudot[c];
         if c=-1 then
         begin
            writecr;
            writetext(‡<BAD‡);
            write(‡ddddd‡,len);
            goto exit
         end;
         A[i]:=integer (((((boolean A[i]) shift -j)
               ∧ 35 m 5 0)
               ∨ boolean c) shift j);
         len:=len+1;
         goto again;
out:     read baudot:=len;
         writecr;
         writetext(‡<Read: ‡);
         writeinteger(‡p‡,len);
      end;
      for i:=0 step 1 until 255 do revbaudot[i]:=-1;
      for i:=0 step 1 until 31 do revbaudot[baudot[i]]:=i;
      case:=0;
      LYN;
      cipherlen:=read baudot(cipher);
      lookforlen:=read baudot(lookfor)
   end;
   for iperm:=1 step 1 until 24 do
   begin
      integer array wheellen[1:5],perm[1:4];
      integer array wheel1,wheel2,wheel3,wheel4,wheel5[0:12];
      integer offset,i;
      boolean procedure genwheels(offset);
      value offset;
      integer offset;
      begin
         integer i,j,c1,c2,b1,b2,b3,b4,b5;
         boolean c3;
         genwheels:=false;
         for i:=0 step 1 until lookforlen-1 do
         begin
            j:=i+offset;
            c1:=fetch char(cipher,j);
            c2:=fetch char(lookfor,i);
            c3:=boolean xor(c1,c2);
            b1:=integer ((c3 ∧ 35 0 5 16)shift -4);
            b2:=integer ((c3 ∧ 35 0 5  8)shift -3);
            b3:=integer ((c3 ∧ 35 0 5  4)shift -2);
            b4:=integer ((c3 ∧ 35 0 5  2)shift -1);
            b5:=integer ((c3 ∧ 35 0 5  1)        );
            if wheel1[j mod wheellen[1]]=-1 then
               wheel1[j mod wheellen[1]] := b1
```

```
            else
               if wheel1[j mod wheellen[1]] ≠ b1 then goto bad;
            if wheel2[j mod wheellen[2]]=-1 then
               wheel2[j mod wheellen[2]] := b2
            else
               if wheel2[j mod wheellen[2]] ≠ b2 then goto bad;
            if wheel3[j mod wheellen[3]]=-1 then
               wheel3[j mod wheellen[3]] := b3
            else
               if wheel3[j mod wheellen[3]] ≠ b3 then goto bad;
            if wheel4[j mod wheellen[4]]=-1 then
               wheel4[j mod wheellen[4]] := b4
            else
               if wheel4[j mod wheellen[4]] ≠ b4 then goto bad;
            if wheel5[j mod wheellen[5]]=-1 then
               wheel5[j mod wheellen[5]] := b5
            else
               if wheel5[j mod wheellen[5]] ≠ b5 then goto bad
         end for i;
         genwheels:=true;
bad:
      end genwheels;
      integer procedure getwheel(offset);
      value offset;
      integer offset;
      getwheel:=
         wheel1[offset mod wheellen[1]]×16 +
         wheel2[offset mod wheellen[2]]× 8 +
         wheel3[offset mod wheellen[3]]× 4 +
         wheel4[offset mod wheellen[4]]× 2 +
         wheel5[offset mod wheellen[5]];
      procedure printclear;
      begin
         integer i,c1,c2,c3,ding,pos,c,clast;
         clast:=-1;
         writecr;
         pos:=0;
         ding:=60;
         for i:=0 step 1 until cipherlen-1 do
         begin
            c1:=fetch char(cipher,i);
            c2:=getwheel(i);
            c3:=xor(c1,c2);
            c:=baudot[c3];
            if clast=-1 then clast:=c
            else
            begin
               if clast≠9 then
               begin
                  WRITE CHAR(clast);
                  pos:=pos+1;
                  clast:=c
               end
               else
               begin
                  if c=9 then
                  begin
                     if pos>ding then
                     begin
                        writecr;
                        pos:=0
                     end CR
                     else
                     begin
```

```
                        writechar(0);
                        pos:=pos+1
                     end space;
                     clast:=-1
                  end
                  else
                  begin
                     WRITE CHAR(clast);
                     pos:=pos+1;
                     clast:=c
                  end
               end
            end
         end for i;
         if clast≠-1 then WRITE CHAR(clast)
      end printclear;
      PERM(perm,4,iperm);
      wheellen[1]:=3;
      for i:=1 step 1 until 4 do
      wheellen[i+1]:=case perm[i] of (5,7,11,13);
      writecr;
      for i:=1 step 1 until 5 do
      writeinteger(⦃ddd⦄,wheellen[i]);
      for offset:=cipherlen-lookforlen step -1 until 0 do
      begin
         for i:=0 step 1 until 12 do
         wheel1[i]:=wheel2[i]:=wheel3[i]:=wheel4[i]:=wheel5[i]:=-1;
         if genwheels(offset) then
         begin
            writecr;
            writechar(58);
            case:=0;
            write(⦃ddddd⦄,wheellen[1],wheellen[2],wheellen[3],wheellen[4],wheellen[5]
            writetext(⦃<  ⦄);
            printclear;
            writechar(58);
            goto done
         end found
      end offset;
   end inner loop;
done:
   writecr;
   writetext(⦃<Time: ⦄);
   write(⦃ddddddd.dd⦄,clock count);
   writecr;
exit:
end;
run<
ANBQVWYFLAK2PJ48N5EU3EGGXVSACBGNZ54RSVW5RM5OFSM4R2W3LL5U95PCZDRUEUBPV2TYKG28WJXRPGCUJ
NORTH99FIFTYFIVE99FORTY
```