Hell's Halls

By Pixelpedant

In search of gold and glory, you travel to a dungeon. It's said incredible wealth hides within the crypts men call Hell's Halls. But evil touches all who enter.

Remain too long, and the dungeon's curse will fall upon you. So claim what wealth you can in the time you have. For with every step you walk, your time grows shorter. In the end, your only hope for victory may lie in finding and slaying the dungeon's mysterious master.

The level loading screen and exploration screens of Hell's Halls. With most of the game spent exploring a dungeon from the overhead view seen in the second image.

TI-99 fans who see *Tunnels of Doom* as the best adventure for the system might take it for granted that you just can't create that kind of magic in TI BASIC. But *Hell's Halls* is one program that means to show adventure games do belong in the BASIC language on the TI-99.

Hell's Halls is compatible with the TI-99/4 and TI-99/4A home computers, but due to its use of the expanded character set of the 99/4A, it cannot be *typed* on the 99/4, and can only run from a saved copy.

If you do not wish to or cannot type the Hell's Halls program, or would like to acquire the complete game package (as illustrated on the facing page), it can be ordered from the author at **https://shop.pixelpedant.com**.

Hell's Halls is played using the ESDX (direction) keys, with which you navigate dangers and collect the treasures found in the game's dungeons.

The game may be saved to cassette or disk, but if a disk controller is installed the **CALL FILES(1)** and **NEW** commands are required on TI BASIC startup to free necessary memory. The typical startup process is then

1) Power on the TI-99 system.
2) Select TI BASIC.
3) Type CALL FILES(1) and press Enter.
4) Type NEW and press Enter.
5) Type the program, or load a stored copy via OLD and press RUN.

Hell's Halls in its cassette release, which includes a second variant of the game on Side B.

1- Startup Sequence

```
1 REM PIXELPEDANT 2022 V2
10 DIM XC(7),YC(5),S(3),EX(4,6),D(4),EE(4,6)
45 GOSUB 5550
50 CALL SCREEN(2)
52 GOSUB 7000
55 GOSUB 5000
64 FOR Q=W TO 1
65 GOSUB 7400
70 FOR N=W TO 3
75 GOSUB 7000
80 GOSUB 2400
81 NEXT N
82 FOR N=W TO 3
84 GOSUB 7400
87 GOSUB 7000
88 NEXT N
89 NEXT Q
90 GOSUB 7400
110 FOR X=W TO 16
115 READ N$
120 PRINT N$
125 NEXT X
```

Author's REMarks

After line 1, REM statements are not used in the program text due to their high memory cost. Instead, consult the pages following for the function of each subprogram.

But for example, 7000 handles CALL CHAR data, while 7400 handles music, and 5000 writes text to screen (also via CALL CHAR, but using a quite different data structure from 7000).

765- Level Setup

```
305 GOSUB 7200
320 GOSUB 7000
325 GOSUB 7000
765 RANDOMIZE
770 FOR X=W TO 3
775 S(X)=W-(X=1)*20
780 NEXT X
785 TH=2
800 RESTORE 9600
805 GOSUB 2298
810 GOSUB 7200
815 GOSUB 7800
820 C=2
825 GOSUB 2330
830 PX=RND*6+.5
835 PY=RND*4+.5
840 EE(PY,PX)=1
845 EX(PY,PX)=1
850 CT=104
855 UE=W
865 ST=W
870 ER=W
```

Author's REMarks

Note that the variable W is undefined, and equates to zero. This use of an undefined W as zero saves 2 bytes on each use over the equivalent constant.

One or two letter variables are a necessary evil here, given 16K TI BASIC memory economy. In a small mercy, these 4 (valid) TI BASIC variable names are unused:] [\ _

890- Dungeon Generation

```
890 RANDOMIZE
895 N$=SEG$("AEKOUZ`AEKPVZ`AGKQV\`AGKQU[`AFLPU[`A
    FLQUZ`AEKPV\`AFKQV\`AFLPV[`",INT(RND*9)*7+1,7)
900 FOR X=1 TO 7
910 GOSUB 7260
915 XC(X)=I
920 NEXT X
960 N$=SEG$("DIMSXDIMRXDJOSXDIOTXDIOSXDHMSXDJNSXDJ
    OTXDINTX",INT(RND*9)*5+1,5)
970 FOR Y=1 TO 5
975 GOSUB 7260
980 YC(Y)=I
990 NEXT Y
1000 GOSUB 2300
1010 PX=XC(PX)+2
1020 PY=YC(PY)+2
1030 IF S(2)>1 THEN 1075
1040 RESTORE 9300
1050 GOSUB 7200
1075 GOSUB 4400
1080 GOSUB 3000
1090 GOSUB 2290
```

1100- Tile-Based Events

```
1100 GOSUB 2470
1120 GOSUB 2235
1130 CALL GCHAR(PY+YM,PX+XM,GC)
1310 ON 339/GC-2 GOTO  1450,1460,1400,
     1100,1500,1400,1430,1650
1320 GOTO 1100
1400 GOSUB 2255
1410 GOTO 1100
1430 GOSUB 7995
1440 GOSUB 2290
1445 GOTO 1100
1450 ON 1815/GC-13 GOTO 2900,1100,2600,
     1400,2730,2700
1460 GOSUB 2290
1465 FOR L=101 TO 95 STEP -1.7
1466 CALL HCHAR(PY+YM,PX+XM,L)
1467 GOSUB 7100
1468 NEXT L
1469 GOSUB 2260
1480 ON RND*2.5+.5 GOSUB 4100,3850,2905
1485 CT=107
1495 GOTO 1100
```

Author's REMarks

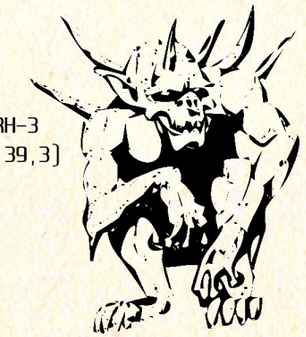
Deciding what to do when the player moves from one tile to another is the most important decision-making process the program will deal with. This needs to be done every time the player provides input, and there are well over a dozen possible outcomes. So an efficient means of identifying these is essential. Did the player walk into a trap? Enter a doorway? Find some gold? We need to know and act accordingly.

The program consequently uses TI BASIC's **ON GOTO** construct to translate tile values to outcomes more rapidly and in a more memory-efficient manner than would a very large series of IF statements.

1500- Door Opening

```
1500 GOSUB 2260
1540 CT=44+(GC=47)*3
1550 GOSUB 1750
1575 GOSUB 4400
1620 IF EX(RY,RX)=1 THEN 1100
1625 ER=ER+1
1630 GOSUB 3000
1635 EX(RY,RX)=1
1640 UE=UE-1
1645 GOTO 1100
1650 CALL HCHAR(PY,PX,CT)
1700 GOSUB 7550
1730 GOTO 800
1740 P=35
1750 FOR X=200 TO 300+P STEP 50-P
1760 CALL SOUND(-X,X,30,X,30,X,30,-4,1)
1770 NEXT X
1780 P=W
1790 RETURN
```

1800- Attacking Imps



```
1800 GOSUB 7995
1830 YM=CY+3-PY
1840 XM=CX+3-PX
1855 FOR Y=CY+2 TO CY+RH-3
1860 CALL HCHAR(Y,CX+2,39,3)
1865 NEXT Y
1870 GOSUB 2255
1880 RESTORE 9410
1885 GOSUB 5075
2000 GOSUB 4500
2015 FOR Q=1 TO 5+S(2)
2020 FOR V=RND+1 TO RND+3.4
2025 IF D(V)=2 THEN 2044
2028 D(V)=D(V)+1
2033 ON V GOTO 2037,2039,2041,2035
2035 CALL HCHAR(PY-3+D(V),PX,125)
2036 GOTO 2042
2037 CALL HCHAR(PY+3-D(V),PX,125)
2038 GOTO 2042
2039 CALL HCHAR(PY,PX-3+D(V),125)
2040 GOTO 2042
2041 CALL HCHAR(PY,PX+3-D(V),125)
2042 IF D(V)<2 THEN 2045
2044 GOSUB 2318
2045 FOR X=W TO 2+(S(2)>1)
2047 CALL KEY(1,K,Z)
2048 IF (K>5)+(K=4)+(ABS(K)=1) THEN 2071
2050 DI=(K+1)/1.5
2051 GOSUB 2290
2052 GOSUB 7995
2053 IF D(DI)<1 THEN 2071
2055 ON DI GOTO 2056,2061,2064,2059
2056 CALL HCHAR(PY+3-D(DI),PX,102)
2057 GOTO 2065
2059 CALL HCHAR(PY-3+D(DI),PX,101)
2060 GOTO 2065
2061 CALL HCHAR(PY,PX-3+D(DI),103)
2062 GOTO 2065
2064 CALL HCHAR(PY,PX+3-D(DI),100)
2065 D(DI)=D(DI)-1
2071 NEXT X
2072 NEXT V
2073 NEXT Q
2075 GOSUB 7600
2080 GOSUB 2285
2090 CT=108
2095 RETURN
```

2235- Player Motion

```
2235 YM=(K=5)-(K<1)
2240 XM=(K=2)-(K=3)
2245 RETURN
2250 GOSUB 2235
2255 CALL SOUND(-120,-5,26)
2260 CALL HCHAR(PY,PX,CT)
2265 PY=PY+YM
2270 PX=PX+XM
2273 S(W)=S(W)+1
2274 IF S(W)<>TH THEN 2284
2275 GOSUB 2285
2276 RESTORE 9420
2277 GOSUB 7400
2278 Q=TH/120
2279 GOSUB 4800
2281 GOSUB 5090
2282 IF Q>4 THEN 2380
2283 TH=TH+120
2284 CT=GC
2285 IF K<W THEN 2286 ELSE 2290
2286 K=W
2290 CALL HCHAR(PY,PX,40+K)
2295 RETURN
```

2298- Various Subprograms

```
2298 N=32
2299 GOTO 2302
2300 N=59
2302 FOR X=4 TO 18 STEP 7
2305 CALL HCHAR(X,1,N,224)
2310 NEXT X
2315 RETURN
2318 C=-1
2320 CALL HCHAR(PY,PX,60)
2325 CALL SOUND(-180,-5,W)
2330 L=(C<W)*2+1
2335 C=ABS(C)
2340 S(C)=S(C)+L
2345 GOSUB 7500
2355 FOR R=1 TO 2
2356 GOSUB 7385
2357 CALL HCHAR(1+R,C,A)
2360 NEXT R
2365 IF L>W THEN 2399
2370 GOSUB 2286
2375 IF S(C)>W THEN 2399
2380 RESTORE 9500
2385 GOSUB 7400
2392 GOSUB 7400
2394 GOSUB 7400
2395 GOSUB 2400
2397 GOTO 7685
2399 RETURN
2400 CALL GCHAR(2,14,X)
2410 X=38-(X=38)*55
2435 CALL HCHAR(2,14,X)
2440 CALL HCHAR(2,20,X)
2450 RETURN
2470 CALL KEY(1,K,Z)
2475 IF (K>5)+(K=4)+(ABS(K)=1) THEN 2470
2480 RETURN
2600 GOSUB 2250
2610 GOSUB 2318
2640 CT=126
2650 RESTORE 9400
2660 GOSUB 5075
2670 GOTO 1100
2700 C=3
2720 GOTO 2750
2730 C=1
2750 GOSUB 2250
2755 CT=10400/GC
2760 GOSUB 7100
2780 GOSUB 2330
2790 GOTO 1100
2900 GOSUB 2250
```

Author's REMarks

Damage, Healing, Gold accumulation and Death outcomes are dealt with in this portion of the program, via player stats contained in the array S.

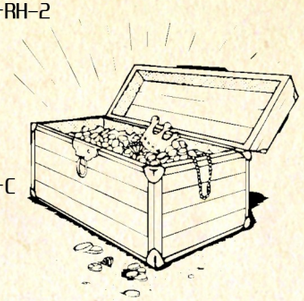
```
2905 GOSUB 4400
2910 FOR Y=1 TO RH-2
2915 CALL HCHAR(CY+Y,CX+1,62,RW-2)
2920 CALL SOUND(-100,-6,Y*2)
2925 GOSUB 2318
2930 NEXT Y
2935 CT=62
2940 GOTO 1100
```

3000- Room Generation

```
3000 FOR X=W TO 3
3005 D(X)=-(RND<0.6^UE+0.4)
3008 NEXT X
3010 IF RY=1 THEN 3020
3015 IF EX(RY-1,RX) THEN 3100
3020 CALL HCHAR(CY,CX,39,RW)
3030 IF (D(W)=W)+(RY=1) THEN 3100
3040 CALL HCHAR(CY,CX+RND*RW/2+1,46)
3050 IF EE(RY-1,RX) THEN 3100
3060 UE=UE+1
3070 EE(RY-1,RX)=1
3100 IF RX=1 THEN 3110
3105 IF EX(RY,RX-1) THEN 3200
3110 CALL VCHAR(CY,CX,39,RH)
3120 IF (D(3)=W)+(RX=1) THEN 3200
3130 CALL HCHAR(CY+RND*RH/2+1,CX,47)
3140 IF EE(RY,RX-1) THEN 3200
3150 UE=UE+1
3160 EE(RY,RX-1)=1
3200 IF RX=6 THEN 3210
3205 IF EX(RY,RX+1) THEN 3300
3210 CALL VCHAR(CY,XC(RX+1),39,RH)
3220 IF (D(1)=W)+(RX=6) THEN 3300
3230 CALL HCHAR(CY+RND*RH/2+1,XC(RX+1),47)
3240 IF EE(RY,RX+1) THEN 3300
3250 UE=UE+1
3260 EE(RY,RX+1)=1
3300 IF RY=4 THEN 3310
3305 IF EX(RY+1,RX) THEN 3400
3310 CALL HCHAR(YC(RY+1),CX,39,RW)
3320 IF (D(2)=W)+(RY=4) THEN 3400
3330 CALL HCHAR(YC(RY+1),CX+RND*RW/2+1,46)
3340 IF EE(RY+1,RX) THEN 3400
3350 UE=UE+1
3360 EE(RY+1,RX)=1
3400 GOSUB 7600
3450 IF ST THEN 3700
3455 IF UE<2 THEN 3480
3470 IF (ER*RND<9) THEN 3700
3480 CALL HCHAR(CY+2,CX+2,34)
3490 ST=-1
3495 RETURN
```

3700- Room-Specific Events

```
3700 IF ER<1 THEN 3495
3705 ON 266/RH/RW-4 GOTO
1800,3800,4000,3950,4100,3680,4300
3800 FOR Y=CY+1 TO CY+RH-2
3810 C=111
3820 GOSUB 4050
3830 NEXT Y
3840 RETURN
3850 C=(RND>0.5)*-2+1
3855 FOR Y=W TO RND*7+C
3860 GOSUB 7100
3865 GOSUB 2330
3870 NEXT Y
3875 C=99-C/3
3880 FOR Y=CY+1 TO CY+RH-1 STEP RH-3
3885 GOSUB 4050
3890 NEXT Y
3895 RETURN
3950 FOR Y=1 TO RND*5+2
3960 CALL HCHAR(CY+RH/2*RND+1,CX+RW/2*RND+1,98)
3970 NEXT Y
3980 RETURN
4000 FOR Y=1 TO RND*4+2
4010 CALL HCHAR(CY+RH/2*RND+1,CX+RW/2*RND+1,99)
4020 NEXT Y
4030 RETURN
4050 XM=RW/2*RND+1
4055 CALL HCHAR(Y,CX+XM,C,RW-XM-1-RND)
4060 RETURN
4100 FOR P=W TO 5*RND+1
4105 GOSUB 4500
4110 FOR B=-1 TO 2 STEP 2
4115 CALL HCHAR(PY+B*YM,PX+B*XM,61)
4120 NEXT B
4130 FOR B=1 TO 3+(S(2)>2)
4135 CALL KEY(1,K,Z)
4140 IF (K<6)*(ABS(K)<>1)*(K<>4) THEN 4150
4145 B=B-.92
4148 GOTO 4185
4150 GOSUB 2285
4155 YO=(K-2)/3*-ABS(YM)
4160 XO=.5/(K-2.5)*ABS(XM)
4165 IF ABS(YO+XO)<.5 THEN 4175
4170 CALL HCHAR(PY+YO,PX+XO,100+(XO-1)*-1.6+YO/2-.4)
4175 GOSUB 7995
4180 D(YO+XO+2)=1
4185 NEXT B
4190 IF D(1)+D(3)=2 THEN 4205
4195 GOSUB 2318
4200 GOTO 4105
4205 NEXT P
4210 FOR B=-1 TO 2 STEP 2
4215 CALL HCHAR(PY+B*YM,PX+B*XM,109)
4220 NEXT B
```



```
4225 GOSUB 2285
4230 RETURN
4300 CALL HCHAR(CY+2,CX+2,96)
4310 RETURN
4400 FOR X=1 TO 7
4405 IF XC(X)>=PX-(XM>W) THEN 4415
4410 NEXT X
4415 RX=X-1
4420 FOR Y=1 TO 5
4425 IF YC(Y)>=PY-(YM>W) THEN 4435
4430 NEXT Y
4435 RY=Y-1
4440 CY=YC(RY)
4445 CX=XC(RX)
4450 RW=XC(RX+1)-CX+1
4455 RH=YC(RY+1)-CY+1
4460 RETURN
4500 FOR Q=W TO 4
4510 D(Q)=W
4520 NEXT Q
4530 RETURN
4800 FOR Y=1 TO Q
4810 READ N$
4820 NEXT Y
4830 RETURN
```

5000- Display At Subprogram

```
5000 READ N$
5005 GOSUB 7260
5010 Y=I
5015 GOSUB 7260
5020 H=ABS(I)
5025 FOR X=W TO LEN(N$)-1
5040 IF I>W THEN 5050
5045 CALL VCHAR(Y-1,H+X,94,3)
5050 GOSUB 7385
5055 IF A>119 THEN 5100
5060 CALL HCHAR(Y,H+X,A)
5065 NEXT X
5070 RETURN
```

5075- Metadata Handling

```
5075 READ F$
5080 IF POS(U$,F$,1) THEN 5070
5085 U$=U$&F$
5090 READ N$
5095 GOSUB 7385
5100 ON A-119 GOTO 7550,5550,5500,7205,5005,7410,7305
5500 FOR P=W TO 500
5510 NEXT P
5520 GOTO 5555
5530 RETURN
5550 CALL CLEAR
5555 IF LEN(N$) THEN 5095
5560 RETURN
```

7000- Call Char Subprogram

```
7000 READ N$
7010 GOSUB 7260
7020 FOR X=1 TO ABS(I)
7025 GOSUB 7385
7030 IF I>W THEN 7055
7035 C=A
7040 GOSUB 7385
7045 CALL CHAR(C,N$)
7055 CALL CHAR(A,N$)
7056 IF X=ABS(I) THEN 7060
7057 READ N$
7060 NEXT X
7065 RETURN
7100 FOR T=220 TO 450 STEP 110
7110 CALL SOUND(-520+T,T,9,T/2,14)
7120 NEXT T
7130 RETURN
```

7200- Call Color Subprogram

```
7200 READ N$
7205 FOR X=1 TO LEN(N$)/3
7215 GOSUB 7260
7220 IF A>119 THEN 5100
7225 D(W)=I
7230 GOSUB 7260
7235 D(1)=I
7240 GOSUB 7260
7245 CALL COLOR(D(W),D(1),I)
7250 NEXT X
7255 RETURN
7260 GOSUB 7385
7265 I=A-64
7270 RETURN
7300 READ N$
7305 FOR X=1 TO LEN(N$)/4+0.75
7310 GOSUB 7385
7315 IF A>119 THEN 5100
7320 D(1)=A
7325 FOR Y=2 TO 4
7330 GOSUB 7260
7335 D(Y)=I
7340 NEXT Y
7345 IF I<W THEN 7360
7350 CALL HCHAR(D(2),D(3),D(1),I)
7355 GOTO 7365
7360 CALL VCHAR(D(2),D(3),D(1),-I)
7365 NEXT X
7370 RETURN
7385 A=ASC(N$)
7390 N$=SEG$(N$,2,LEN(N$)-1)
7395 RETURN
```

7400- Music Subprogram

```
7400 READ N$
7410 GOSUB 7260
7420 FOR X=1 TO LEN(N$)
7425 GOSUB 7385
7430 IF A>119 THEN 5100
7435 A=(A*A*A/14400+25)*4
7440 FOR V=11 TO I+15 STEP 50/I
7445 CALL SOUND(-425*I,A,V,A+2,V,A*3.81,30,-4,V-I)
7450 NEXT V
7455 NEXT X
7460 RETURN
7500 N$=STR$(S(C))
7510 IF S(C)>9 THEN 7530
7520 N$="0"&N$
7530 RETURN
7550 N$="^AJU^BJU^CJU"&N$
7560 GOTO 7305
7600 FOR Q=1 TO RH-2
7605 CALL HCHAR(CY+Q,CX+1,104,RW-2)
7610 NEXT Q
7615 RETURN
7650 GOSUB 5000
7655 GOSUB 7300
7660 CALL CLEAR
7665 GOSUB 7200
7670 GOSUB 7780
7675 GOSUB 7780
7685 RESTORE 9550
7690 GOSUB 5090
7700 RESTORE 9140
7705 CALL KEY(W,C,Q)
7710 IF Q=W THEN 7705
7715 GOSUB 5550
7720 GOTO 110
7780 READ C
7781 GOSUB 7500
7782 N$=SEG$("NYHY",C/2+1,2)&N$
7783 GOSUB 5005
7784 GOSUB 5000
7785 RETURN
7800 Q=S(2)
7810 GOSUB 4800
7820 GOSUB 5090
7835 IF Q=4 THEN 7650
7845 FOR L=W TO 6
7850 GOSUB 8000
7855 FOR Y=1 TO 4
7860 EE(Y,L)=SIN(W)
7865 EX(Y,L)=W
7870 NEXT Y
7875 NEXT L
7880 GOSUB 1740
7885 RETURN
7995 L=W
8000 CALL SOUND(-110,-7,L)
8010 RETURN
```


8700- Data Statements

This portion of the program includes nearly all graphical and sound data used by Hell's Halls. This begins with pattern data for the game's opening sequence, including its main font.

```
8700 DATA "G!030C304080601807","#C0300C02010618E","$000000186600018E7","%E7180066l8"
8720 DATA "&FF7E000000007EFF","]3C7EFFE7E7FF7E3C","^0008002A0008"
8760 DATA "BM!]#   !]#|AN$     $|CN%     %{EGBFGBGGBAOBHGB","J7:747:747"
8800 DATA "JA0018242432C2442", "B0038242834243B", "C0018242020241B", "D00582424242458","E005C2038202052C"
8810 DATA "F005C2020382O2", "G003844404C4438","H004224243C2466","I007C101010107C", "J0018080808483"
8920 DATA "DK00485060504848", "L004040404040078","M00446C54444444", "N0042223222A642","JO003844444444438"
8930 DATA "P0038444870404","Q003048484850528", "R0038444870484B", "S003C42300C423C","T0038541010103B"
8940 DATA "U0084484848483","V0044444444281", "W0044444444542B","Y0044282810101","D{5E525E4CBE4D1E33"
8950 DATA "]A57E0000007EA5FF","*1C9CCC7E2D0C1E33","+3839337EB43078CC"
9000 DATA "JA7|G;WHO ARE YOU","C,C3A5E7A5A5E7A5C3","-7A7A7A327DB278CC",".FF99DB9999DB99FF"
9010 DATA "J>F|J:WHO WANDERS","C/99DB999999DB99FF","000182424242418","100081808080B1C"
9020 DATA "JAL|M;IN MY TOWER","C2001824040810 3C","3001824080424l8","40008182828 3C08"
9030 DATA "JF>|P:AT MY WALLS","C6000810382424l8","7003C040810101","8001824l8242418"
9040 DATA "J@7","B;FFFFE7FBF7FFF7FF","<3C24BD5A3C183C66","Bp83C7E3C183C7E3F1","q03070F0703070F07"
9050 DATA "BrC0E0F0E0C0E0F0E","x00286D6D7D6D6D","Cy0000D111D111DD","z00000C100C02DC","n0"
9080 DATA "JF>|G-GO AND FLEE","C|002A55AA55AA54","5003C2038042418","_0038440408100 1"
9090 DATA "J7>|J,LEST YOU BE","Ca0006061E1E7E7E","9001824241C081","{0000D151D1515D"
9100 DATA "J4!|M-FOREVERMORE","Cd000020FEFC2","e00101818183C1818","h0C122121225E9088"
9110 DATA "JA>7|P,LOST WITHIN","Bf18183C18181808","g0000047F3F04","J4>Fy"
9140 DATA "bpxyzpr","0rsssqr","0px{zpr",
9145 DATA "   h   h hhhh h   h h hhhh    h   h h    h   h   h       hhhhh hhhh h   h   hhhh"
9150 DATA "   h   h h    h   h     h    h   h hhhh hhh hhh hhhh"
9155 DATA ,,,"   h   h hhhh h   h   hhhh    h   h h  h h   h   h       hhhhh hhhh h   h   hhhh"
9160 DATA "   h   h h  h h   h     h    h   h h  h hhh hhh hhhh",,,,,,
9145 DATA "   h   h hhhh h   h h hhhh    h   h h    h   h   h       hhhhh hhhh h   h   hhhh"
9150 DATA "   h   h h    h   h     h    h   h hhhh hhh hhh hhhh"
9155 DATA ,,,"   h   h hhhh h   h   hhhh    h   h h  h h   h   h       hhhhh hhhh h   h   hhhh"
9160 DATA "   h   h h  h h   h     h    h   h h  h hhh hhh hhhh",,,,,,
9165 DATA "JGE}JA>7A>74>7'!47~cAAAaABA2BAA0BBA0CAA1CBAqA_=rA^="
9170 DATA "F""0001010505151555","'FBFBFB00DFDFDF","Z0010101010001","[0000000018081","@0008101"
9175 DATA ">1002401100842108","9ci005A187E7E185A","bj3078360F36783","^k3C665A7A161C141C"
9180 DATA "}1423C5A3C5AA53C66","=m00245AFFDB81","~o003C5A66665A3C","\sFFFFEE44002267FF"
9300 DATA "HBBx{IKELIEKEBHGBCBEDBEJFEBBE"
9400 DATA 1,"}JF:>x|AJHAHZ|BJEVEN MY FLOORS|CJDEFY YOU"
9410 DATA 2,"}JF>FC:C>x|AJTRAPPED_|BJWELL DONE[ MY IMPSZ"
9420 DATA "JCA0!'","x|AJFOUR HOURS UNTIL|BJIF YOU WANDER STILL|CJALL IS LOST"
9430 DATA "x|AJTHREE HOURS YET|BJUNTIL YOUR DOOM"
9440 DATA "x|AJTWO HOURS NOW UNTIL|BJMY CURSE IS CAST"
9450 DATA "x|AJTHREE HOURS SPENT|BJAND BUT ONE REMAINS"
9460 DATA "x|AJWITH FOUR HOURS GONE|BJYOUR FATE IS SEALED"
9500 DATA "JCA:7:::7:y{HBBABB|AN$     $|BM!]#   !]#|CN%  $  %|EO\\\\\{AOBHGB"
9510 DATA "JN|GLAS YOU FALL}JC|HLFOOLISH FOE}JI|JLI TO YOU MY}JA|KLGEAS BESTOW}JC|MLBINDING YOU"
9520 DATA "J:|NLTO MY WALLS}JA|PLFOREVERMORE}J7|RLPRISONER TO}J:|TLHELLS HALLS}Jc^[YNC:7:"
9550 DATA "zzy{ABBJBBCBBDBBIBBKBBLBB|J6^PRESS ANY KEY|N6^TO PLAY AGAIN^"
9600 DATA "ABB}JA~'DA^}J7~'RA^}J>~'TKL}J4~'VJN}J7~'XIP}J'~'EL3}J7~'EU3"
9605 DATA "|HM&&#  !&&|NM&&#  !&&{AOB|SMYOU FIND|UJTHE FRONT GATE|WIAND SNEAK INSIDEz"
9610 DATA "|GM%#|GS!%|HM$#|HS!$|NM%#|NS!%|OM$#|OS!${AKB|SKSTEPS ASCEND|UKTO A DOOR OF|WKSHINING GOLDz"
9615 DATA "~%ENF$QNF#EM3!ET3{AGB|SKTHIS DOORWAY|UMBEARS AN|WKEERIE SCRAWLz|HPNO|IOTIME|KPNO|LNESCAPEz"
9620 DATA "|JS$|KS&|LS%{AFB|SLWEAKLY LIT|UJBY A BLUE GLOW|WIIS A FINAL FLOORz"
9625 DATA "|MP$$|OO!&&#|PO!  #|QO!  #|NP]]{AMB|SLA HUGE BUG|ULBLOCKS YOU|WLAND SHOUTS"
9650 DATA "FO AS |GNMASTER|HOHERE|JMI DEMAND|KMYOU STOPz"
9660 DATA " SKL UJN WIP|SKYOU SLAY HIM|NP^^|ULAND REVEAL|WLA PATH OUT~qMO;nMP;nMQ;rMR;}FS^^ksnf^^zz"
9670 DATA "CBEDBE|B6^IN THE HALLS|HHYOU GAIN IN GOLD",3,"NHWALKING IN PACES",0,"U7^WELL DONE[ HEROz"
```

Top TI Type-in Tip: Keep in mind that any typographical errors introduced in transcribing the above data will have a high likelihood of crashing the program with an out-of-range value. The user is therefore encouraged to regard typing these 4,346 characters wholly without error as a fun and entertaining challenge.