SOURCE CODE:

ORG 100H

.MODEL SMALL

.STACK 100H

;-------------------------DATA AND VARIABLES-------------------------

;we use "db" (data byte) to allocate some space, and fill it with a string

.DATA

WELCOME\_MSG DB "================================================================================",

DB " ++++++++++++++++++++++ WELCOME TO +++++++++++++++++++++ ",

DB " ++++++++++++++++++++++ MAZE PATH FINDER GAME +++++++++++++++++++++ ",

DB "================================================================================",

CREATED\_BY DB " CREATED BY :- ", 0

CODER DB " ", 0

DB " ", 0

DB " ", 0

DB " \*\*\* ALI GAUHAR 02134202-006 \*\*\* ", 0

DB " ", 0

DB " \*\*\* NAEEM UR REHMAN 02-134202-053 \*\*\* ", 0

DB " ", 0

DB " \*\*\* NAMEER 02-134172-172 \*\*\* ", 0

DB " ", 0

PROMPT DB "INPUT TO MOVE NEXT...",

SELECT\_INS DB "PRESS 1 TO PLAY THE GAME, ",0

DB "PRESS 2 TO SEE THE INSTRUCTION, ",0

DB "PRESS 3 TO SEE THE SCORES, ",0

DB "PRESS 4 TO EXIT. ",0

DB " Waiting For Key ... ",0

MAIN\_MENU DB " ", 0

DB " \*\*\*\*\*\*\*\*\*\*\* START GAME \*\*\*\*\*\*\*\*\*\*\*\* ", 0

DB " ", 0

DB " \*\*\*\*\*\*\*\*\*\*\* INSTRUCTIONS \*\*\*\*\*\*\*\*\*\*\*\* ", 0

DB " ", 0

DB " \*\*\*\*\*\*\*\*\*\*\* HIGH SCORES \*\*\*\*\*\*\*\*\*\*\*\* ", 0

DB " ", 0

DB " \*\*\*\*\*\*\*\*\*\*\* QUIT GAME \*\*\*\*\*\*\*\*\*\*\*\* ", 0

DB " ", 0

INSTRUCTS DB "===================================================",0

DB "| | ",0

DB "| Find The Path With In MAX Moves Given And | ",0

DB "| Navigate With Minimum Moves To Score Extra. | ",0

DB "| Use Arow Keys To Control Directions And | ",0

DB "| Use 'escape key' To Quit Any Time. | ",0

DB "| | ",0

DB "===================================================",0

GREETINGS DB " ", 0

DB " \*\*\*\* !!CONGRATULATIONSS!! \*\*\*\* ", 0

DB " ", 0

SCORE\_NOTE DB " YOUR SCORES: "

TOTAL\_POINT DW '0'

GAME\_POINT DW '0'

GAME\_GONE DB " ", 0

DB " \*\*\*\*\*\* !!!GAME OVER!!! \*\*\*\*\*\*\* ", 0

DB " ", 0

CONGO\_EXIT DB " ", 0

DB " \*\* !!!THANKS FOR PLAYING!!! \*\* ", 0

DB " ", 0

DB " TOTAL SCORES: "

EXIT\_NOTE DB "Do You Want TO quit ?", 0

DB "Press Enter Key To Quit", 0

P1\_COUNT DB '0'

P2\_COUNT DB '0'

P3\_COUNT DB '0'

;storing in intiger formate

BOOLEAN DB 0

COUNT\_MAX DW 0

LINE\_NO DB 0

LEVEL\_CHK DB 0

EXIST\_CHK DB 0

;--------------------------------------------------------------------

;-------------------------CODE AREA----------------------------------

.CODE

;++++++++++++++++++++++++++++ MAIN METHOD +++++++++++++++++++++++++++

MAIN PROC

;way of preparing the data to be printed...pointing to the data

MOV AX, @DATA ; It moves memory location of data to AX

MOV DS, AX ; It moves location in AX to DS

;calling the function defined below

CALL RESET ; To Call RESET Method

CALL HEAD ; To Call HEAD Method

CALL NAMES ; To Call NAMES Method

CALL INPUT ; To Call INPUT Method

CALL RESET

CALL HEAD

;setting the the pointer to print the menu in the center

MOV DL, 35 ; Set cursor position Column

MOV DH, 24 ; Set cursor position Row

CALL MENU ; To Call MENU Method

;INT 10h / AH = 0Ah - write character only at cursor position.

;03h - text mode. 80x25. 16 colors. 8 pages.

;to get out of the game mode and turn the screen black

MOV AH, 0 ; To set video mode

MOV AL, 3 ; To set which video mode (text mode, 80x25, 16 colors, 8 pages)

INT 10H ; Interrupt to initialize video mode

;returning control to the dos

MOV AH, 4CH ; To set program return the controls

INT 21H ; Call Interrupt to return controls

MAIN ENDP

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

;++++++++++++++++++++++++++++ RESET METHOD ++++++++++++++++++++++++++

RESET PROC

;function to prepare the console for game

;To push back the previous values of the registers inorder to save new valuses

PUSH AX

PUSH BX

PUSH CX

PUSH DX

;AH=00h->AL = video mode

;this is to set video mode

;(AH=02h) BH = Page Number, DH = Row, DL = Column

MOV AH, 0

MOV AL, 2

INT 10H

; Scrolls a specified window upward a specified number of lines.

;

; On entry: AH 06h

; AL Number of lines to scroll (if 0, clear entire

; window)

; BH Display attribute for blank lines

; CH Row number of upper left corner

; CL Column number of upper left corner

; DH Row number of lower right corner

; DL Column number of lower right corner

; Returns: None

; Registers destroyed: AX, SP, BP, SI, DI

Mov AH, 06H ; To scroll up the window

MOV AL, 0 ; To set number of lines to scrolled (0 = to clear entire window)

MOV BH, 11100000B ; To write blank lines at bottom of window (Here we have given color>

MOV CH, 0 ; Upper row

MOV CL, 0 ; Upper Column

MOV DI, 1 ; Rows on screen -1,

MOV DH, [DI] ; Lower Row

MOV DI, 0 ; Columns on screen

MOV DL, [DI] ; Lower Column

INT 10H ; Call interrupt for video mode

;seting back the previous values as our work has been done

POP DX ; Pop previous DX value from stack

POP CX ; Pop previous CX value from stack

POP BX ; Pop previous BX value from stack

POP AX ; Pop previous AX value from stack

RET

RESET ENDP

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

;++++++++++++++++++++++++++++ HEAD METHOD ++++++++++++++++++++++++++

HEAD PROC

;for taking new values

PUSH AX

PUSH CX

;fetching the first line of welcome message

;LEA->load address of the string

;MOV AX, 0B80Ah<- it divides the current line at which it is standing into 10 pieces and gives spaces according to the input... here the 0B8 is the opcode

;and the next 2 values are used for giving the space converted into hexadecimal

MOV AX, 0B800h ; Set a displacement from current position

LEA SI, WELCOME\_MSG ; memory address of first character of welcoming message

MOV CX, 80 ; setting the counter equal to the emu width

CALL DISP ; To display the holded string

;fetching the second line of welcome message

MOV AX, 0B800h

LEA SI, WELCOME\_MSG+80

MOV CX, 80

CALL DISP

;fetching the third line of welcome message

MOV AX, 0B800h

LEA SI, WELCOME\_MSG+160

MOV CX, 80

CALL DISP

;fetching the fourth line of welcome message

MOV AX, 0B800h

LEA SI, WELCOME\_MSG+240

MOV CX, 80

CALL DISP

;again reseting the values of the registers

POP CX

POP AX

RET;end of function

HEAD ENDP

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

;+++++++++++++++++++++++++++++ DISP METHOD ++++++++++++++++++++++++++

DISP PROC ;performs the display operation

;pushing back the AX values

PUSH AX

;saving ax to extra segment to use ah again

MOV ES, AX ; Moving data in AX to Extra Segment

MOV AH, 31 ; To set the text color

;looping to print the string

;The LODS instruction loads the value from the corresponding locations pointed by

;DS:SI into the accumulator (AL, AX, or EAX) and then increments (or decrements) SI by one, two

;The STOS instruction copies the data item from AL (for words - STOSW)

DO:

LODSB ; It loads unit byte pointed by SI from the memory

STOSW ; STOSW Copy AX to memory

LOOP DO ; Loop until cx becomes 0

;again reseting the registers

POP AX

RET

DISP ENDP

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

;+++++++++++++++++++++++++++ NAMES METHOD +++++++++++++++++++++++++++

NAMES PROC

;pushing back the registers value

PUSH AX

PUSH CX

;to print created by

MOV AX, 0B817H ; Set a displacement from current position

LEA SI, CREATED\_BY ; To provide memory address of first character or array

MOV CX, 24 ; To set count (used to set how many times a loop execute>

CALL DISP ; To call DISP method

MOV AX, 0B81EH

LEA SI, CREATED\_BY+18

MOV CX, 24

CALL DISP

MOV AX, 0B825H

LEA SI, CREATED\_BY

MOV CX, 24

CALL DISP

;to print the names of the coder

MOV AX, 0B835H

LEA SI, CODER+98

MOV CX, 48

CALL DISP

MOV AX, 0B839H

LEA SI, CODER+147

MOV CX, 48

CALL DISP

MOV AX, 0B83DH

LEA SI, CODER+195

MOV CX, 48

CALL DISP

MOV AX, 0B841H

LEA SI, CODER+245

MOV CX, 48

CALL DISP

MOV AX, 0B845H

LEA SI, CODER+294

MOV CX, 48

CALL DISP

MOV AX, 0B849H

LEA SI, CODER+343

MOV CX, 48

CALL DISP

MOV AX, 0B84DH

LEA SI, CODER+294

MOV CX, 48

CALL DISP

;to ask for input

MOV AX, 0B88DH

LEA SI, PROMPT

MOV CX, 21

CALL DISP

;reseting the registers

POP CX

POP AX

RET

NAMES ENDP

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

;++++++++++++++++++++++++++++ INPUT METHOD ++++++++++++++++++++++++++

INPUT PROC

;pushing back the values in registers

PUSH AX

PUSH BX

PUSH DX

MOV BH, 0 ; To point towards current page

MOV DL, 34 ; To which column to give offset to the cursor

MOV DH, 24 ; To which row to give offset to the cursor

MOV AH, 02 ; To change cursor position

INT 10H ; Interrupt to enter into graphical mode

;to take input from the keyboard

MOV AH, 0 ; Get keystroke from keyboard (no echo).

INT 16H ; Interrupt prompts for keybord input input

;again reseting the registers

POP DX

POP BX

POP AX

RET

INPUT ENDP

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

;++++++++++++++++++++++++++++ MENU METHOD +++++++++++++++++++++++++++

MENU PROC

;this is for showing menu

;pushing back the previous values

PUSH AX

PUSH CX

;same methods as for the aboves

MOV AX, 0B82AH ; Set a displacement from current position

LEA SI, MAIN\_MENU ; To provide memory address of first character or array

MOV CX, 48 ; To set count (used to set how many times a loop execute>

CALL DISP ; To call DISP method

MOV AX, 0B82EH

LEA SI, MAIN\_MENU+49

MOV CX, 48

CALL DISP

MOV AX, 0B832H

LEA SI, MAIN\_MENU+98

MOV CX, 48

CALL DISP

MOV AX, 0B836H

LEA SI, MAIN\_MENU+147

MOV CX, 48

CALL DISP

MOV AX, 0B83AH

LEA SI, MAIN\_MENU+195

MOV CX, 48

CALL DISP

MOV AX, 0B83EH

LEA SI, MAIN\_MENU+245

MOV CX, 48

CALL DISP

MOV AX, 0B842H

LEA SI, MAIN\_MENU+294

MOV CX, 48

CALL DISP

MOV AX, 0B846H

LEA SI, MAIN\_MENU+343

MOV CX, 48

CALL DISP

MOV AX, 0B84AH

LEA SI, MAIN\_MENU+294

MOV CX, 48

CALL DISP

;menu shown, now for the selection from the menu

CALL SELECT

;reseting the values

POP CX

POP AX

RET

MENU ENDP

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

;+++++++++++++++++++++++++++ SELECT METHOD ++++++++++++++++++++++++++

SELECT PROC

;selection instructions printing

MOV AX, 0B856H ;cursor position setting

LEA SI, SELECT\_INS

MOV CX, 32

CALL DISP

MOV AX, 0B85CH

LEA SI, SELECT\_INS+34

MOV CX, 32

CALL DISP

MOV AX, 0B862H

LEA SI, SELECT\_INS+68

MOV CX, 32

CALL DISP

MOV AX, 0B868H

LEA SI, SELECT\_INS+102

MOV CX, 32

CALL DISP

MOV AX, 0B878H

LEA SI, SELECT\_INS+134

MOV CX, 32

CALL DISP

;pointing on the screen

;AH=02h BH = Page Number, DH = Row, DL = Column

MOV AH, 2 ; To set Cursor Position

MOV BH, 0 ; To set Page Number

INT 10H ; Intrupt

;AH = Scan code of the key pressed down

;AL = ASCII character of the button pressed

MOV AH, 00h ; Get keystroke from keyboard (no echo)

INT 16H ; Interrupt to get from keyboard

DEC AH ;... here we are decrementing the input because the emulator is getting 2 when hitting 1

;comparing the inputs and calling the functions according to it

;cmp is the comlparing statement

;jpm is jump condition according to the condition given

CMP AH, 01 ; Comparing AH with 01

JE STRT\_GAME ; If above comparison is Equal then jump to STRT\_GAME

CMP AH, 02 ; Comparing AH with 02

JE INSTRUCTIONS ; If above comparison is Equal then jump to INSTRUCTIONS

CMP AH, 03 ; Comparing AH with 03

JE SCORE\_JUMP ; If above comparison is Equal then jump to SCORE\_JUMP

CMP AH, 04 ; Comparing AH with 04

JE QUIT ; If above comparison is Equal then jump to QUIT

JMP EXIT\_OUT ; If above comparison is Equal to none then jump to EXIT\_OUT

;calling functions for start game instruction on 1

STRT\_GAME:

CALL LEVEL1 ; Call LEVEL1 Methods (drawing the map)

CALL GAME1 ; Call GAME1 Methods (actual controll and game)

CALL RESET

CALL HEAD

CALL MENU

CALL SELECT

;calling functions for instruction check on 2

INSTRUCTIONS:

CALL RESET

CALL HEAD

MOV AX, 0B834H

LEA SI, INSTRUCTS

MOV CX, 48

CALL DISP

MOV AX, 0B838H

LEA SI, INSTRUCTS+52

MOV CX, 48

CALL DISP

MOV AX, 0B83CH

LEA SI, INSTRUCTS+104

MOV CX, 48

CALL DISP

MOV AX, 0B840H

LEA SI, INSTRUCTS+156

MOV CX, 48

CALL DISP

MOV AX, 0B844H

LEA SI, INSTRUCTS+208

MOV CX, 48

CALL DISP

MOV AX, 0B848H

LEA SI, INSTRUCTS+260

MOV CX, 48

CALL DISP

MOV AX, 0B84CH

LEA SI, INSTRUCTS+312

MOV CX, 48

CALL DISP

MOV AX, 0B850H

LEA SI, INSTRUCTS+364

MOV CX, 48

CALL DISP

MOV AX, 0B88EH

LEA SI, PROMPT

MOV CX, 21

CALL DISP

CALL INPUT

JMP EXIT\_OUT

;calling functions for high score instruction on 3

SCORE\_JUMP:

CALL RESET

CALL HEAD

MOV AX, 0B835H

LEA SI, SCORE\_NOTE

MOV CX, 16

CALL DISP

;////////////////////// intention to convert byte into double word because the score are based on 2 digits

PUSH AX

PUSH DX

PUSH BX

SUB GAME\_POINT, 48 ; Converting number from ascii 1 to decimal 0

MOV AX, GAME\_POINT ; Points will be in AX

MOV CX, 10 ; we will divide points by 10

CWD ; Convert Word to Double Word

DIV CX ; This will divide value in AX with CX

ADD DX, 48 ; Adding 48 to make it ascii 0

PUSH DX ; Push it to the stack

CWD ; Convert Word to Double Word

DIV CX ; This will divide value in AX with CX

ADD DX, 48 ; Adding 48 to make it ascii 0

PUSH DX ; Push it to the stack

MOV BH, 0 ; To point towards current page

MOV CH, 41 ; To initialize counter

myLoop:

MOV DL, CH ; To which column to give offset to the cursor

MOV DH, 10 ; To which row to give offset to the cursor

MOV AH, 02 ; To change cursor position

INT 10H ; Interrupt to enter into graphical mode

MOV AH, 2 ; To write a character to standard output

POP DX ; Pop value from stack to DX

INT 21H ; Interrup to call to text mode

INC CH ; Increment counter

CMP CH, 43 ; Comparing counter with 43

JNE myLoop ; Jump to myLoop if above comparison is not equal

MOV CH, 0 ; Moving 0 to CH

POP BX

POP DX

POP AX

;//////////////////////////

MOV AX, 0B835H

LEA SI, GAME\_POINT

MOV CX, 2

CALL DISP

MOV AX, 0B8BCH

LEA SI, PROMPT

MOV CX, 21

CALL DISP

CALL INPUT

ADD GAME\_POINT, 48 ; Again converting GAME\_POINT to ascii 0

JMP EXIT\_OUT ; Jump to EXIT\_OUT

;calling functions for exit instruction on 4...this will take you to the menu

EXIT\_OUT:

CALL RESET

CALL HEAD

CALL MENU

;

QUIT:

CALL RESET

JMP OK

OK:

RET

SELECT ENDP

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

;+++++++++++++++++++++++++++ LEVEL1 METHOD ++++++++++++++++++++++++++

LEVEL1 PROC

MOV AH, 0 ; Set video mode

MOV AL, 13H ; To set graphical mode, 40x25, 256 colors.

INT 10H ; Interrupt to set video mode

;to change the pixels color green in order to print the walls

;the walls conprrison with the 5\*5 box will be detected through the color comparison

MOV AH, 0CH ; change color for a single pixel

MOV AL, 10 ; to set pixel color

MOV CX, 0

MOV DX, 0

MOV BX, 0

;printing the horizontal line at pixel row no 0 and 199, the couner is 320 as the height is of 320 pixels

;int 10h is changing the pixel color as defined above

HORIZONTAL: ; To draw Horizontal Border

MOV DX, 10 ; Row

INT 10H ; Interrupt

MOV DX, 199

INT 10H

INC CX

CMP CX, 320 ;since the

JNE HORIZONTAL

;printing the verticle line at pixel cloumn 0 and 319, the counter is 199 as the width is 200

;int 10h is changing the pixel of the color as defined above

MOV DX, 0

VERTICLE: ; To draw Verticle Border

MOV CX, 0

INT 10H

MOV CX, 319

INT 10H

INC DX

CMP DX, 199

JNE VERTICLE

;printing the horiczontal line at (21th pixels after) from the top (0-300)<- to left 30 pixel of the end free for the box to pass from it

;;l1

MOV CX, 0

LINE\_1:

MOV DX, 21

INT 10H

INC CX

CMP CX, 300

JNE LINE\_1

;printing the horiczontal line at (32th pixels after) from the top ,width is 70 because we have to give some space for the box to move from it

;the horizontal distance from the previous line is given 32-21=11 pixels

;;l2

MOV CX, 0

LINE\_2A:

MOV DX, 32

INT 10H

INC CX

CMP CX, 70

JNE LINE\_2A

;gap of 16 is given between the previous and next horizontal line

ADD CX, 16

;same technique is used as above

LINE\_2B:

MOV DX, 32

INT 10H

INC CX

CMP CX, 300

JNE LINE\_2B

;same technique is used as above

;this is the third line as dx is incremented to 11

;width is 35

;;l3

MOV CX, 16;started from column 16

LINE\_3A:

MOV DX, 43

INT 10H

INC CX

CMP CX, 35

JNE LINE\_3A

;same technique is used as above ...continue line

ADD CX, 16

LINE\_3B:

MOV DX, 43

INT 10H

INC CX

CMP CX, 120

JNE LINE\_3B

;same technique is used as above ...continue line

ADD CX, 16

LINE\_3C:

MOV DX, 43

INT 10H

INC CX

CMP CX, 300

JNE LINE\_3C

;same technique is used as above

;;l4

MOV CX, 16

LINE\_4A:

MOV DX, 54

INT 10H

INC CX

CMP CX, 100

JNE LINE\_4A

;same technique is used as above ...continue line

ADD CX, 16

LINE\_4B:

MOV DX, 54

INT 10H

INC CX

CMP CX, 300

JNE LINE\_4B

;same technique is used as above

;;l5

MOV CX, 0

LINE\_5A:

MOV DX, 65

INT 10H

INC CX

CMP CX, 50

JNE LINE\_5A

;same technique is used as above ...continue line

ADD CX, 16

LINE\_5B:

MOV DX, 65

INT 10H

INC CX

CMP CX, 300

JNE LINE\_5B

;same technique is used as above

;;l6

MOV CX, 16

LINE\_6:

MOV DX, 76

INT 10H

INC CX

CMP CX, 319

JNE LINE\_6

;same technique is used as above

;;l7

MOV CX, 16

LINE\_7A:

MOV DX, 87

INT 10H

INC CX

CMP CX, 70

JNE LINE\_7A

;same technique is used as above ...continue line

ADD CX, 16

LINE\_7B:

MOV DX, 87

INT 10H

INC CX

CMP CX, 300

JNE LINE\_7B

;same technique is used as above

;;l8

MOV CX, 0

LINE\_8A:

MOV DX, 98

INT 10H

INC CX

CMP CX, 270

JNE LINE\_8A

;same technique is used as above ...continue line

ADD CX, 16

LINE\_8B:

MOV DX, 98

INT 10H

INC CX

CMP CX, 300

JNE LINE\_8B

;same technique is used as above

;;l9

MOV CX, 0

LINE\_9A:

MOV DX, 109

INT 10H

INC CX

CMP CX, 20

JNE LINE\_9A

;same technique is used as above ...continue line

ADD CX, 16

LINE\_9B:

MOV DX, 109

INT 10H

INC CX

CMP CX, 285

JNE LINE\_9B

;same technique is used as above

;;l10

MOV CX, 0

LINE\_10A:

MOV DX, 120

INT 10H

INC CX

CMP CX, 70

JNE LINE\_10A

;same technique is used as above ...continue line

ADD CX, 16

LINE\_10B:

MOV DX, 120

INT 10H

INC CX

CMP CX, 300

JNE LINE\_10B

;same technique is used as above

;;L11

MOV CX, 0

LINE\_11A:

MOV DX, 131

INT 10H

INC CX

CMP CX, 55

JNE LINE\_11A

;same technique is used as above ...continue line

ADD CX, 16

LINE\_11B:

MOV DX, 131

INT 10H

INC CX

CMP CX, 100

JNE LINE\_11B

;same technique is used as above ...continue line

ADD CX, 16

LINE\_11C:

MOV DX, 131

INT 10H

INC CX

CMP CX, 319

JNE LINE\_11C

;same technique is used as above

;;L12

MOV CX, 16

LINE\_12A:

MOV DX, 142

INT 10H

INC CX

CMP CX, 80

JNE LINE\_12A

;same technique is used as above ...continue line

ADD CX, 16

LINE\_12B:

MOV DX, 142

INT 10H

INC CX

CMP CX, 260

JNE LINE\_12B

;same technique is used as above ...continue line

ADD CX, 16

LINE\_12C:

MOV DX, 142

INT 10H

INC CX

CMP CX, 319

JNE LINE\_12C

;same technique is used as above

;;L13

MOV CX, 16

LINE\_13A:

MOV DX, 153

INT 10H

INC CX

CMP CX, 70

JNE LINE\_13A

;same technique is used as above ...continue line

ADD CX, 16

LINE\_13B:

MOV DX, 153

INT 10H

INC CX

CMP CX, 300

JNE LINE\_13B

;same technique is used as above

;;l14

MOV CX, 16

LINE\_14A:

MOV DX, 164

INT 10H

INC CX

CMP CX, 70

JNE LINE\_14A

;same technique is used as above ...continue line

ADD CX, 16

LINE\_14B:

MOV DX, 164

INT 10H

INC CX

CMP CX, 170

JNE LINE\_14B

;same technique is used as above ...continue line

ADD CX, 16

LINE\_14C:

MOV DX, 164

INT 10H

INC CX

CMP CX, 300

JNE LINE\_14C

;same technique is used as above

;;l15

MOV CX, 16

LINE\_15A:

MOV DX, 175

INT 10H

INC CX

CMP CX, 70

JNE LINE\_15A

;same technique is used as above ...continue line

ADD CX, 16

LINE\_15B:

MOV DX, 175

INT 10H

INC CX

CMP CX, 255

JNE LINE\_15B

;same technique is used as above ...continue line

ADD CX, 16

LINE\_15C:

MOV DX, 175

INT 10H

INC CX

CMP CX, 319

JNE LINE\_15C

;same technique is used as above

;;L16

MOV CX, 16

LINE\_16:

MOV DX, 186

INT 10H

INC CX

CMP CX, 270

JNE LINE\_16

;same technique is used as above

;;l17

MOV CX, 0

LINE\_17:

MOV DX, 197

INT 10H

INC CX

CMP CX, 319

JNE LINE\_17

; verticle printing starts from here..................

; cx is used as column selector

; dx is used as row selector

;;v1

MOV CX, 300

MOV DX, 21 ;from 21 to 32 ... verticle height was given 11 above

VERTICLE\_0:

INT 10h

INC DX

CMP DX, 32

JNE VERTICLE\_0

;same technique is used as above

;;v2

MOV DX, 43

VERTICLE\_1:

INT 10h

INC DX

CMP DX, 55

JNE VERTICLE\_1

;same technique is used as above

;;v3

MOV DX, 87

VERTICLE\_2:

INT 10h

INC DX

CMP DX, 120

JNE VERTICLE\_2

;same technique is used as above................dealing with cx=30 till here

;;v4

MOV DX, 32

MOV CX, 90

VERTICLE\_3:

INT 10h

INC DX

CMP DX, 43

JNE VERTICLE\_3

;same technique is used as above

;;v5

MOV DX, 43

VERTICLE\_4:

MOV CX, 35

INT 10h

MOV CX, 120

INT 10h

INC DX

CMP DX, 54

JNE VERTICLE\_4

;same technique is used as above

;;v6

VERTICLE\_5:

MOV CX, 200

INT 10H

INC DX

CMP DX, 65

JNE VERTICLE\_5

;same technique is used as above

;;v7

MOV DX, 76

VERTICLE\_6:

MOV CX, 35

INT 10h

MOV CX, 90

INT 10h

INC DX

CMP DX, 87

JNE VERTICLE\_6

;same technique is used as above

;;v8

MOV DX, 109

VERTICLE\_7:

MOV CX, 70

INT 10h

INC DX

CMP DX, 120

JNE VERTICLE\_7

;same technique is used as above

;;v9

MOV DX, 131

VERTICLE\_8:

MOV CX, 80

INT 10h

MOV CX, 100

INT 10h

INC DX

CMP DX, 142

JNE VERTICLE\_8

;same technique is used as above

;;v10

MOV DX, 153

VERTICLE\_9:

MOV CX, 70

INT 10h

MOV CX, 90

INT 10h

MOV CX, 255

INT 10h

INC DX

CMP DX, 164

JNE VERTICLE\_9

;same technique is used as above

;;v11

VERTICLE\_10:

MOV CX, 18

INT 10h

MOV CX, 255

INT 10h

INC DX

CMP DX, 175

JNE VERTICLE\_10

;same technique is used as above

;;v12

MOV DX, 142

VERTICLE\_11:

MOV CX, 20

INT 10h

INC DX

CMP DX, 153

JNE VERTICLE\_11

;same technique is used as above

;;v13

MOV DX, 175

VERTICLE\_12:

MOV CX, 160

INT 10H

INC DX

CMP DX, 186

JNE VERTICLE\_12

;same technique is used as above

;;v14

VERTICLE\_13:

MOV CX, 180

INT 10H

INC DX

CMP DX, 197

JNE VERTICLE\_13

;this is the goal line of yellow color AL=14

;;Ending Line

MOV CX, 305

MOV DX, 197

MOV AL, 14

END\_20:

MOV AH, 0CH

INT 10h

DEC DX

CMP DX, 175

JA END\_20

RET

LEVEL1 ENDP

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

;++++++++++++++++++++++++++ GAME1 METHOD ++++++++++++++++++++++++++++

GAME1 PROC

;this is the main procedure of the game

MOV DX, 11 ; the above 10 pixels is out of the boundary therefore starting printing the box from 11 Row

MOV CX, 1 ; since there is no width out of the horizonal wall therefore Column=1

MOV BX, 0505H ; box of 5\*5 al and ah are used for representing

DRW:

CMP COUNT\_MAX, 500 ; count max is the variable counts the moves and overs the game at 500 moves

JNE CON ; Loops for 500 moves if not equal to 500 moves

CALL GAME\_OVER ; function to terminate the game

JMP EXI ; Jump to F

CON: ; if moves are not equal to 500 then execute the function as given sequentially

MOV AX, 0C0FH ; To change the color of single pixel

CALL DISPLAY ; Call DISPLAY method

CALL KBHIT ; Call KBHIT method

JMP DRW ; Jump to DRW

EXI:

RET

GAME1 ENDP

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

;+++++++++++++++++++++++++++ DISPLAY METHOD +++++++++++++++++++++++++

DISPLAY PROC ;function to print the box

PUSH AX

PUSH CX

PUSH DX

PUSH BX

D1:

PUSH CX

D2: ;cx is specifying the position to be printed and bl is for specifying the width

INT 10h ; Interrupt to print colored pixel->white color for box->(defined in the game1 label CON)

INC CX ; Increment column->specifying the position from current to next 5 of the ball

DEC BL ; Decrementing BL -> BL=5->defined in the game1,it is widtth of the ball

CMP BL, 0 ; Checking if BL is 0

JNE D2 ; If above comparison is not equal then jump D2

;bh is for specifying the height->reducing to print to next line

DEC BH ; Decrementing BH

INC DX ; Incrementing DX

POP CX ; POP CX->again refillint the width

MOV BL, 5 ; Moving 5 to BL->again refilling the width for next line

CMP BH, 0 ; Checking if BH is 0

JNE D1 ; If above comparison is not equal then jump D1

POP BX

POP DX

POP CX

POP AX

RET

DISPLAY ENDP

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

;++++++++++++++++++++++++ GAME\_OVER METHOD ++++++++++++++++++++++++++

GAME\_OVER PROC

CALL RESET ; Call RESET Method

CALL HEAD ; Call HEAD Method

;game over quotes

MOV AX, 0B82BH ; To give offset to text on screen

LEA SI, GAME\_GONE ; Move offset of variable in SI

MOV CX, 32 ; Initialize counter with 16

CALL DISP ; Call DISP Method

MOV AX, 0B831H

LEA SI, GAME\_GONE+33

MOV CX, 32

CALL DISP

MOV AX, 0B837H

LEA SI, GAME\_GONE+66

MOV CX, 32

CALL DISP

;showing scores

MOV AX, 0B852H ; To give offset to text on screen

LEA SI, SCORE\_NOTE ; Move offset of variable in SI

MOV CX, 16 ; Initialize counter with 14

CALL DISP ; Call DISP Method

ADD COUNT\_MAX, 48; converting ascii into string formate

MOV AX, 0B852H

LEA SI, COUNT\_MAX

MOV CX, 2

CALL DISP

MOV AX, 0B8B0H

LEA SI, PROMPT

MOV CX, 21

CALL DISP

RET

GAME\_OVER ENDP

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

;++++++++++++++++++++++++++ KBHIT METHOD ++++++++++++++++++++++++++++

KBHIT proc

;procedure to get input from the keyboard

MOV AH, 00h ; get keystroke from keyboard (no echo).

INT 16h ; Waits for user input

;asii value of up arrow is 72->comparing with 72 to proceed for up direction

CMP AH, 72 ; Comparing AX with 72

JZ UP1 ; Jump to UP1 if above comparison is equal

;asii value of up arrow is 80->comparing with 72 to proceed for down direction

CMP AH, 80 ; Comparing AX with 80

JZ DOWN1 ; Jump to DOWN1 if above comparison is equal

;asii value of up arrow is 75->comparing with 72 to proceed for left direction

CMP AH, 75 ; Comparing AX with 75

JZ LEFT ; Jump to LEFT if above comparison is equal

;asii value of up arrow is 77->comparing with 72 to proceed for right direction

CMP AH, 77 ; Comparing AX with 77

JZ RIGHT ; Jump to RIGHT if above comparison is equal

;if any other keyis pressed then exit

JMP EXIT ; Else exit

;if upper key is pressed then decrement (dx->representing the top) means move up

;then call the check method to check weather the pixels going to collide with the upper wall are of green color? if yes? then stop the process

UP1:

PUSH DX

SUB DX, 1 ; Subtract DX to move row up

CALL CHECK ; Call CHECK Method

CMP BOOLEAN, 1 ; To check if line is being crossed or not

POP DX

JE EXIT ; Jump to EXIT if ball tries to moves through the line

;again changing color because it was changed in the check label for comparison

MOV AX, 0C00H ; Subroutine to change color of single pixel

CALL DISPLAY ; Call DISPLAY Method

DEC DX ; Decrementing DX

CALL LINE ; Call LINE Method

SUB DX, 4 ; Subtracting 4 from DX to move 4 rows above

CMP LINE\_NO, 1 ; Comparing LINE\_NO with 1

JNE EXIT ; Jump to EXIT if comparison is not equal

MOV LINE\_NO, 0 ; Moving 0 to LINE\_NO

DEC DX ; Decrementing DX

CALL COUNTER ; Call COUNTER Method

JMP EXIT ; Jump to EXIT

DOWN1:

PUSH DX

ADD DX, 5 ; Add 5 to DX to move 5 row down

CALL CHECK ; Call CHECK Method

CMP BOOLEAN, 1 ; To check if line is being crossed or not

POP DX

JE EXIT ; Jump to EXIT if ball tries to moves through the line

MOV AX, 0C00H ; Subroutine to change color of single pixel

CALL DISPLAY ; Call DISPLAY Method

ADD DX, 5 ; Adding 5 to DX

CALL LINE ; Call LINE Method

CMP LINE\_NO, 1 ; Comparing LINE\_NO with 1

JNE EXIT ; Jump to EXIT if comparison is not equal

MOV LINE\_NO, 0 ; Moving 0 to LINE\_NO

INC DX ; Incrementing DX

CALL COUNTER ; Call COUNTER Method

JMP EXIT ; Jump to EXIT

LEFT:

PUSH CX

SUB CX, 1 ; Subtract 1 from CX to move 1 column left

CALL CHECK ; Call CHECK method

CMP BOOLEAN, 1 ; To check if line is being crossed or not

POP CX

JE EXIT ; Jump to EXIT if ball tries to moves through the line

MOV AX, 0C00H ; Subroutine to change color of single pixel

CALL DISPLAY ; Call DISPLAY Method

SUB CX, 5 ; Subtract 5 from CX

CALL COUNTER ; Call COUNTER Method

JMP EXIT ; Jump to EXIT

RIGHT:

PUSH CX

ADD CX, 9 ; Add 9 to CX to move 9 columns right

CALL CHECK ; Call CHECK Method

CMP BOOLEAN, 1 ; To check if line is being crossed or not

POP CX

JE EXIT ; Jump to EXIT if ball tries to moves through the line

;watch for end line if not then continue

CMP AL, 14 ; CMP for exit line with colour 14

JNE FFF ; Jump to FFF if above comparison is not equal

CALL CONGRETS ; Call CONGRETS method

JMP EXIT ; Jump to EXIT

FFF:

MOV AX, 0C00H ; changing the color to black inorder ro delete the previous box and calling the display method

CALL DISPLAY ; Call DISPLAY method

ADD CX, 5 ; to increment the box size inn order ro be displaced from its position

CALL COUNTER ; Call COUNTER method for changing the move rate

EXIT:

MOV BOOLEAN, 0 ; Resetting the BOOLEAN

RET

KBHIT ENDP

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

;+++++++++++++++++++++++++++ CHECK METHOD ++++++++++++++++++++++++++++

CHECK proc

;the method of checking will only be used for left and right move

PUSH BX

;taking the pixel of the cursor to check if the color is green->means wall

MOV AH, 0DH ; To get color of a single pixel

MOV BH, 0 ; Page 0

INT 10H ; Interrupt to get color

;checking if the color is green?

CMP AL, 10 ; Compare current pixel color with present pixel color

JNE EX ; Jump to EX if above comparison is equal

MOV BOOLEAN, 1 ; Else mov 1 to BOOLEAN

EX:

POP BX

RET

CHECK endp

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

;+++++++++++++++++++++++++++ LINE METHOD ++++++++++++++++++++++++++++

LINE PROC

;the method of comparing the defined lines with the upper and lower boundary is only for up and down move

;following are the defined lines to be compaired

CMP DX, 21 ; Comparing DX with 21 because at row 21 there is our first line

JE KL ; Jump to KL if our comparison is equal

CMP DX, 32

JE KL

CMP DX, 43

JE KL

CMP DX, 54

JE KL

CMP DX, 65

JE KL

CMP DX, 76

JE KL

CMP DX, 87

JE KL

CMP DX, 98

JE KL

CMP DX, 109

JE KL

CMP DX, 120

JE KL

CMP DX, 131

JE KL

CMP DX, 142

JE KL

CMP DX, 153

JE KL

CMP DX, 164

JE KL

CMP DX, 175

JE KL

CMP DX, 186

JNE EF ; Else Jump to EF

KL:

MOV LINE\_NO, 1 ; Moving 1 to LINE\_NO -> working as a boolean

EF:

RET

LINE ENDP

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

;+++++++++++++++++++++++++ COUNTER METHOD +++++++++++++++++++++++++++

COUNTER PROC

;this method counts the moves for the box

PUSH CX

PUSH DX

PUSH BX

;printing the counts

MOV AH, 2 ; Subroutine to write character to standard output

;Counters to store the scores into string formate

;p1->denoting the first digit of the counter

;p2->denoting the second digit of the counter

;p3->denoting the third digit of the counter

MOV DL, P1\_COUNT ; Moving P1\_COUNT to DL

INT 21h ; Interrupt to display character

MOV DL, P2\_COUNT ; Moving P2\_COUNT to DL

INT 21h ; Interrupt to display character

MOV DL, P3\_COUNT ; Moving P3\_COUNT to DL

INT 21h ; Interrupt to display character

;setting cursor for the printing the box in further moves

MOV DL, 0 ; Move Cursor to Column 1

MOV DH, 0 ; Move Cursor to Row 1

MOV AH, 02 ; Subroutine to change cursor position

MOV BH, 0 ; Mov to page 0

INT 10h ; Interrupt to set cursor position

;increment in left most counter if it is not equal to 9

;if equal to 9 then make it 0 and increment in second counter

CMP P3\_COUNT, '9' ; Comparing P3\_COUNT with ascii 9

JE NEX1 ; Jump to NEX1 if above comparison is equal

INC P3\_COUNT ; Incrementing P3\_COUNT

JMP SEC ; Jump to SEC

;same procedure for next1

NEX1:

CMP P2\_COUNT, '9' ; Comparing P2\_COUNT with ascii 9

JE NEX2 ; Jump to NEX2 if above comparison is equal

MOV P3\_COUNT, '0' ; Comparing P2\_COUNT with ascii 0

INC P2\_COUNT ; Incrementing P2\_COUNT

JMP SEC ; Jump to SEC

;same procedure for next2

NEX2:

INC P1\_COUNT ; Incrementing P1\_COUNT

MOV P2\_COUNT, '0' ; Moving ascii 0 to P2\_COUNT

MOV P3\_COUNT, '0' ; Moving ascii 0 to P3\_COUNT

;incrementing the moves in count\_max

SEC:

ADD COUNT\_MAX, 1 ; Add 1 to COUNT\_MAX

POP BX

POP DX

POP CX

RET

COUNTER ENDP

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

;++++++++++++++++++++++++ CONGRETS METHOD +++++++++++++++++++++++++++

CONGRETS PROC ;calling in case of game winning

PUSH CX

CALL RESET ; Call RESET Method

CALL HEAD ; Call HEAD Method

;specifying the position of player according to the moves->less moves more scores

CMP COUNT\_MAX, 390 ; Comparing COUNT\_MAX to 390

JB FIRST ; Jump to FIRST if COUNT\_MAX is less than 390

CMP COUNT\_MAX, 420 ; Comparing COUNT\_MAX to 420

JB SECOND ; Jump to SECOND if COUNT\_MAX is less than 420

CMP COUNT\_MAX, 450 ; Comparing COUNT\_MAX to 450

JB THIRD ; Jump to THIRD if COUNT\_MAX is less than 450

;if more then 450 then points are 3

MOV GAME\_POINT, '3' ; Else move 3 to GAME\_POINT

JMP CONTINUE ; Jump to CONTINUE

;specifying the points of the player

FIRST:

MOV GAME\_POINT, '9' ; Moving 9 to GAME\_POINT

JMP CONTINUE ; Jump to CONTINUE

SECOND:

MOV GAME\_POINT, '7' ; Moving 7 to GAME\_POINT

JMP CONTINUE ; Jump to CONTINUE

THIRD:

MOV GAME\_POINT, '5' ; Moving 5 to GAME\_POINT

;different type of points calculation for moves more then 450

PUSH CX

PUSH DX

MOV CX, GAME\_POINT ; Moving GAME\_POINT to CX

MOV DX, TOTAL\_POINT ; Moving TOTAL\_POINT to DX

MOV GAME\_POINT, DX ; Moving TOTAL\_POINT from DX to GAME\_POINT

MOV TOTAL\_POINT, CX ; Moving GAME\_POINT from CX to TOTAL\_POINT

POP DX

POP CX

;printing the greetings

CONTINUE:

MOV AX, 0B817H ; To give offset to text on screen

LEA SI, GREETINGS ; Move offset of variable in SI

MOV CX, 32 ; Initialize counter with 16

CALL DISP ; Call DISP Method

MOV AX, 0B81DH

LEA SI, GREETINGS+33

MOV CX, 32

CALL DISP

MOV AX, 0B823H

LEA SI, GREETINGS+66

MOV CX, 32

CALL DISP

;printing the scores and maximum counts

MOV AX, 0B833H ; To give offset to text on screen

LEA SI, SCORE\_NOTE ; Move offset of variable in SI

MOV CX, 20 ; Initialize counter with 14

CALL DISP ; Call DISP Method

ADD COUNT\_MAX, 48

MOV AX, 0B833H

LEA SI, COUNT\_MAX

MOV CX, 4

CALL DISP

;printing the ending lines

MOV AX, 0B844H

LEA SI, CONGO\_EXIT

MOV CX, 32

CALL DISP

MOV AX, 0B84AH

LEA SI, CONGO\_EXIT+33

MOV CX, 32

CALL DISP

MOV AX, 0B850H

LEA SI, CONGO\_EXIT+66

MOV CX, 32

CALL DISP

MOV AX, 0B860H

LEA SI, CONGO\_EXIT+99

MOV CX, 18

CALL DISP

;converting the ascii into the numeric form

;calculating total points by adding remaining moves into the scores

SUB TOTAL\_POINT, 48 ; Subtracting 48 from TOTAL\_POINT

MOV CX, TOTAL\_POINT ; Moving TOTAL\_POINT to CX

SUB GAME\_POINT , 48 ; Subtracting 48 from GAME\_POINT

ADD GAME\_POINT , CX ; Add GAME\_POINT to CX->to specify the game points according to the moves left

PUSH DX

PUSH BX

;converting the formate to perform the operations to specify the game points

MOV AX, GAME\_POINT ; Moving GAME\_POINT to AX

MOV CX, 10 ; Initializing counter with 10

CWD ; Convert Word to Byte

DIV CX ; Divide value in AX from CX

ADD DX, 48 ; Add 48 in DX which has remainder to convert it into ascii number

PUSH DX ; Push DX to stack

CWD ; Convert Word to Byte

DIV CX

ADD DX, 48

PUSH DX

MOV BH, 0 ; To point towards current page

MOV CH, 43 ; To initialize counter

myLoop2:;for poping the required dx value

;specify the position to start print

MOV DL, CH ; To which column to give offset to the cursor

MOV DH, 17 ; To which row to give offset to the cursor

MOV AH, 02 ; To change cursor position

INT 10H ; Interrupt to enter into graphical mode

;printing the scores

MOV AH, 2 ; To write a character to standard output

POP DX ; Pop value from stack to DX

INT 21H ; Interrup to call to text mode

INC CH ; Increment counter

CMP CH, 45 ; Comparing counter with 43

JNE myLoop2 ; Jump to myLoop if above comparison is not equal

MOV CH, 0 ; Moving 0 to CH

POP BX

POP DX

POP AX

;asking for input

MOV AX, 0B8A1H

LEA SI, PROMPT

MOV CX, 21

CALL DISP

CALL INPUT

POP BX

POP DX

POP CX

RET

CONGRETS ENDP

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++