```
; DFS program example
; Eliav Gnessin, Summer 2002
; ===============================================
; This is an example program in 8086 Assembly
; ===============================================

TITLE DFS

; This instruction defines the memory model that MASM or TASM use
.model small

; Define the stack size. This instruction initializes the SP
.stack 160h

; Variables & other definitions section
.data
arg     dw 10            ; this is the output base for printax
zero    dw 0

; declare tree
; -----------------------------------------
;                       9
;                      / \
;                    /     \
;                  /         \
;                7             8
;              /   \         /   \
;            3       4     5       6
;          /                         \
;        1                             2
;
; -----------------------------------------

T1      db 1
        dw -1
        dw -1
T2      db 2
        dw -1
        dw -1
T3      db 3
        dw T1
        dw -1
T4      db 4
        dw -1
        dw -1
T5      db 5
        dw -1
        dw -1
T6      db 6
        dw -1
        dw T2
T7      db 7
        dw T3
        dw T4
T8      db 8
        dw T5
        dw T6
T9      db 9    ; tree head
        dw T7
        dw T8

; This is the program itself
.code
start:  mov ax,@data        ; Since the .data instruction doesn't initialize
        mov ds,ax           ; the ds register we have to do it manually

        lea bx,T9           ; get our tree pointer
        call dfsrec         ; compute

        mov ax,4c00h        ; This is the program terminator
        int 21h             ; just like putting "return 0" in C

; =================================================
; Procedure definitions
; =================================================

; =======================================================
; Procedure name: dfsrec - recursive Depth-First-Search
; Input:          BX - pointer to tree node
; Output:         None, only prints node and DFS(node)
; =======================================================
dfsrec    proc near
          mov al,[bx]
          mov ah,0
          call printax;
          mov ax,-1
          cmp ax,[bx+1]         ; halt condition
          jne re_call1          ; if not - do recursion
sec_comp: mov ax,-1
          cmp ax,[bx+3]         ; halt condition
          jne re_call2          ; if not - do recursion
          jmp done
re_call1: push bx               ; put parameter in stack
          mov bx,[bx+1]
          call dfsrec           ; recursive call
          pop bx
          jmp sec_comp          ; don't forget to check right node
re_call2: push bx               ; put parameter in stack
          mov bx,[bx+3]
          call dfsrec           ; recursive call
          pop bx
done:     ret
dfsrec    endp


; =======================================================
; Procedure name: printax - print AX register in base arg
; Input:          AX - the number to be printed
;                 arg - output base [2-10]
; Output:         None
; =======================================================
printax proc near
        push bx
        mov si,0            ; si will count the num of digits
again4: mov dx,0
        div arg             ; AX/arg-> reminder is in DX
        add dx,30h          ; convert value to ASCII: 0-9 => "0"-"9"
        push dx             ; Store in stack
        inc si
        cmp zero,ax         ; if the quotient is 0, we are finished
        mov cx,2            ; make sure the loop doesn't finish because
                            ;  CX=0
        loopnz again4

        ; Move down to next line - Carriage Return + Line Feed
        mov cx,si           ; CX will count the result's digits

        mov al,10           ; Print CR + LF
        call printch
        mov al,13
        call printch

again5: pop ax              ; get result from stack and print it
        call printch
        loop again5
        pop bx
        ret
printax endp

; =================================================
; Procedure name: printch - Print a char to console
; Input:          AL - the char's ASCII code
; Output:         None
; =================================================
printch proc near
        mov bx,0            ; No color definitions
        mov ah,0Eh          ; Print char to TTY function code
        int 10h             ; Call
        ret
printch endp

; End of program
end start
```