N EQU 61 ; # cells

M EQU 3 ; # players

WGF_RACE DB N DUP(?) → Record: | Ø Ø Ø Ø C C C C |

C → Award/Penalty code (if any)
C → Ø Ø Ø Ø (NO AWARD / PENALTY)

61 cells → $2^6 = 64$ → 6 bits to represent the position of each player

PLAYERS DB 3 DUP(?)
→ Record: | 1/Ø 1/Ø P P P P P P |

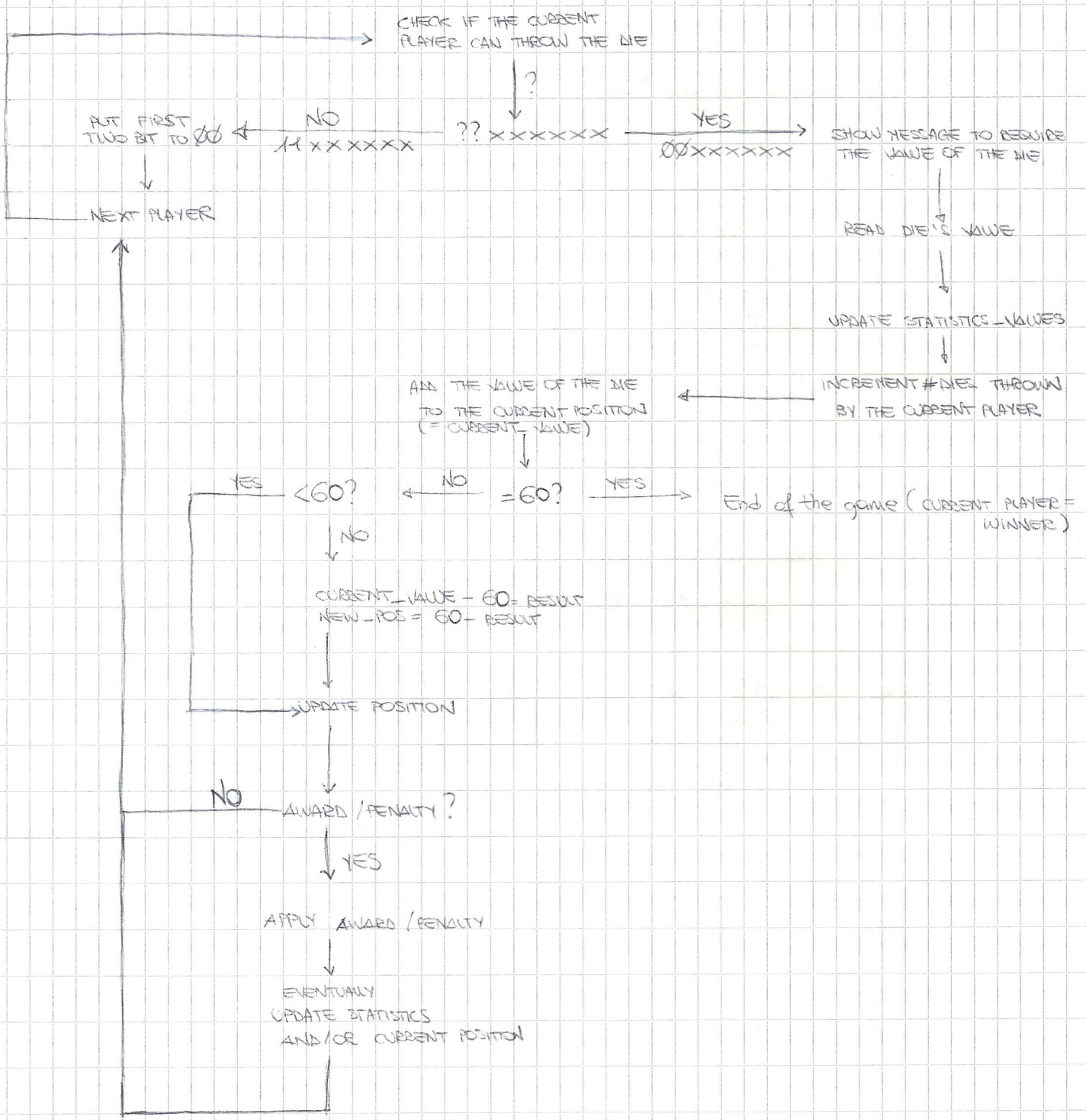
→ P = position (initialized to Ø for each player)
→ 1 if this player cannot throw the die at the next round (penalty)
Ø otherwise (initial value)

STATISTICS_VALUES DB 6 DUP (Ø)
→ #1, #2, #3, ..., #6 thrown during the whole game

STATISTICS_PLAYERS DB 3*2 DUP (Ø)
→ entry: [Ø] [1]
→ one entry for each player

[Ø] → # STEPS DEPENDING ON AWARD / PENALTIES
[1] → # DIES THROWN BY THE PLAYER

AT EACH ROUND...



```
; Implement game and relative statistics
; Awards and penalties are implemented by using a JUMP TABLE in which each position (cell) calls a
  procedure managing the corresponding award/penalty (only if position P is 1≤P≤7)
JT   DW  A1, A2, A3, P1, P2, P3, P4
DIE  DB  0


  xor si, si
gameloop:
    mov CL, PLAYERS[si]
    test CL, 11000000B
    jnz missTurn
    ; show message to require the die's value
    mov AH, 1      ; read the value of the die
    int 21H
    sub AL, 0      ; AL <- VALUE OF THE DIE
    mov DIE, AL    ; to be used after (just in case)

    MOV DL, AL          ; update statistics: count of values
    Xor DH, DH
    DEC DX
    mov DI, DX
    add STATISTICS_VALUES[di], 1

    xor di, di          ; increment #dies thrown by the current player
    add di, si
    add di, di
    add STATISTICS_PLAYERS[di+1], 1

    add CL, AL          ; compute the "new position"
    cmp CL, 60
    je endOfTheGame
```

```
    mov AL, CL
    cmp AL, 60
    jl updatePosition      ;if nextPositin < 60 ==> updatePosition
    mov al, 60             ; otherwise, next position > 60 ==> move backward
    SUB CL, AL             ; next position - 60 = Value (>0)
    SUB AL, CL             ; AL <== ACTUAL NEW POSITION
updatePosition:
    mov PLAYERS[si], AL
    cmp AL, 7
    jg noPenaltyOrAward
    ; otherwise print that the current cell has an award or a penalty
    xor BH, BH             ;start the code relative to a penalty/award
    mov BL, PLAYERS[si]
    SUB BX, 1
    ADD BX, BX
    jmp WORD ptr JT[bx]

missTurn:
    and PLAYERS[si], 00111111B
nextPlayer:
    ; print current position of the current player
    cmp si, 2
    je restartPlayers
    add si, 1
    jmp gameloop
restartPlayers:
    mov si, 0
    jmp nextPlayer
noPenaltyOrAward:
    ; print msg no penalty or award
    jmp nextPlayer
endOfTheGame:
    ; print Winner (current player)
    jmp exit
```