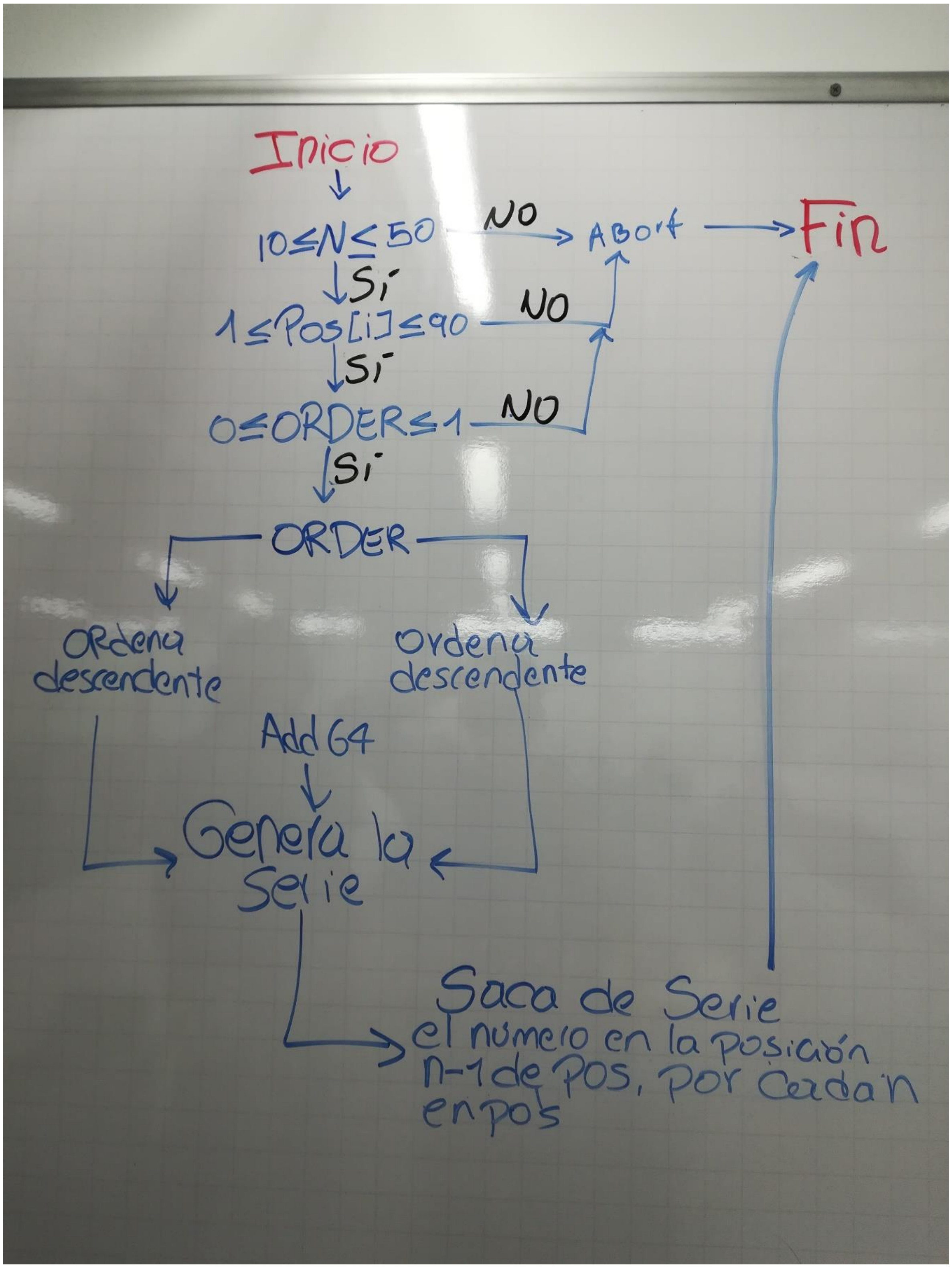
```
}

// MAIN

main(){

// Verificaciones del programa

if N < 10 || N > 50
        leave()
for i=0; i < N; i++
        if (POS[i]<1 || POS[i]>90)
                leave()
if ORDER!=1 || ORDER!=0
        leave()
leave(){
        SortedValues[0] = A5A5A5A5
        exit()
}

//funciones de ordenamiento, serie fibonacci, guardado de los valores
Sort(N, POS, ORDER)
Fibonacci(*SERIE, SIZE)
Guardar(*SERIE, N, *POS, *Sortedvalues)
Final()
}
Final(){
        Final()
}
```

```
Sort(N, *POS, ORDER){
for i=0; i<N-1; i++{
        for j=0; j<N-1-i; j++{
                if !ORDER{
                        if POS[j]>POS[j+1]{
                                temp=POS[j]
                                POS[j]=POS[j+1]
                                POS[j+1]=temp


                        }
                }
        }
}
}


Fibonacci(*SERIE, SIZE){
SERIE[0]=0
SERIE[1]=0
SERIE[2]=1
SERIE[3]=0


for i=4; i<SIZE*2;i+=2{
        MSB_B = SERIE[i-1]
        LSB_B = SERIE[i-2]
        MSB_A = SERIE[i-3]
        LSB_A = SERIE[i-4]


        MSB_R,LSB_R = Add64(MSB_A,LSB_A,MSB_B,LSB_B)
        SERIE[i] = LSB_R

        SERIE[i+1] = MSB_R


        }

}


Add64(A2,A1, B2, B1){

        R1 = A1 + B1   //----> Genera carry C

        R2 = A2 + B2 + C

        return R2, R1

}

Guardar(*SERIE, N, *POS, *SortedValues){

for (i=0; i<N*2; i+=2){

        aux=POS[i]

        SortedValues[i]=SERIE[(aux-1)*2]

        Sortedvalues[i+1]=SERIE[((aux-1)*2)+1]

}

return

}
```


Inicio
10≤N≤50
NO
Abort
Fin
Si
1≤Pos[i]≤90
NO
Si
0≤ORDER≤1
NO
Si
ORDER
Ordena descendente
Ordena descendente
Add 64
Genera la Serie
Saca de Serie el numero en la posición n-1 de Pos, por cada n en pos