

Aclaracion: Cuando decimes esHeap(...) nos referimos a que el observador heap que es array cumple el orden correspondiente. El orden que tiene que tener esta aclarado en el invariante de maxHeap al final de todo.

```

InvRepSistemaCNE(s :SistemaCNE){
    [sinRepetidos(s.nombresPartidos) &&L s.nombresPartidos[-1] = "Blancos" ]
    &&L
    [ sinRepetidos(s.nombresDistritos) &&L |s.nombresDistritos| = |s.diputadosPorDistrito| ]
    &&L
    [ |s.diputadosPorDistrito| = | s.nombresDistrito | ]
    &&L
    [(|s.rangoMesasDistritosInf| = |s.nombresDistritos|) &&L (s.rangoMesasDistritosInf[0] =
    0) &&L ( (forall i : Z)( 0 <= i < |s.rangoMesasDistritosInf| -1 )
    →(s.rangoMesasDistritosSup[i] = s.rangoMesasDistritosInf[i+1] ) ) &&L ((forall i : Z)( 0 <=
    i < |s.rangoMesasDistritosInf| ) →(s.rangoMesasDistritosInf[i] <=
    s.rangoMesasDistritosSup[i]) ) &&L estaOrdenada(s.rangoMesasDistritosInf) ]
    &&L
    [ (|s.rangoMesasDistritosSup| = |s.nombresDistritos|) &&L ( s.nombresDistritos[-1] =
    |mesasRegistradas| ) ]
    &&L
    [ |s.votosPresidenciales| = |s.nombresPartidos| ]
    &&L
    [(|s.votosPresidencialesHeap| = |s.nombresPartidos| - 1) &&L
    esHeap(s.votosPresidencialHeap) ]
    &&L
    [|s.votosDiputados| = |s.nombresDistritos| &&L ( (forall i : Z)(0 <= i < |s.votosDiputados|)
    →(s.votosDiputados[i] = |s.nombresPartidos|) ) ]
    &&L
    [ |s.votosDiputadosXDistHeap| = |s.nombresDistritos| &&L ( (forall i : Z)(0 <= i <
    |s.nombresDistritos|) →(|s.votosDiputadosXDistHeap[i]| = |s.nombresPartidos| - 1))
    &&L [ Si s.calcDip[i] = False entonces s.votosDiputadosXDistHeap[i] tiene los mismos
    elementos que votos diputados (no necesariamente en el mismo orden) y vale
    esHeap(s.votosDiputadosXDistHeap[i]), si s.calcDip[i] = True entonces (forall j:Z)(
    0<=j<|s.nombresPartidos|-1 implicaLuego s.votosDiputados[i][j]/(s.resDip[i][j] + 1) es elemento
    de s.votosDiputadosCDistHeap[i].heap) &&L esHeap(s.votosDiputadosCDistHeap[i]) ]
    &&L
    [|s.mesasRegistradas| = s.rangoMesasDistritoSup[-1]]
    &&L
    [s.totalVotos = suma(s.votosPresidenciales) ]
    &&L
    [(|s.totalVotosoDist| = |s.nombresDistritos|) &&L ( (forall i : Z)(0<= i <
    |s.nombresDistritos|) → s.totalVotosDist[i] = suma(votosDiputados[i] ) ) ]
    &&L

```

```

    [(|s.calcDip| = |s.nombresDistritos|) &&L ( si calcDip[i] = True entonces suma(s.resDip[i]
= s.diputadosPorDistritos[i] y ademas votosDiputados[i][j]/(resDip[i][j] + 1) es elemento de
s.votosDiputadosXDistHeap[i].heap donde j esta en rango de la cantidad de partidos, si
calcDip[i] = False s.resDip[i] es un array ceros de largo |s.nombresPartidos| - 1 y ademas los
elementos de s.votosDiputados[i] son los mismos que s.votosDipXDistHeap[i].heap) ]

```

```

    &&L

```

```

    [(|s.resDip| = |s.nombres.Distrito|) &&L (Para las i's en rango |s.resDip[i] =
|s.nombresPartido|-1) &&L (si calcDip[i] = True entonces suma(s.resDip[i] =
diputadosPorDistrito[i] y ademas resDip[i][j] indica la cantidad de diputados del partido i ej el
distrito j , si no calcDip[i]= False entonces resDip[i]=[0,...,0])
}

```

```

InvRepHeap(h : maxHeap){

```

```

    (h.cota >= 0) &&L (h.size >= 0) &&L (h.cota>= h.size) &&L (|h.heap| = size) &&L ((forall i
:Z)(0<= i <= h.size/2-1)--> (h.heap[i]>=h.heap(2*i +1) && h.heap[i]>=h.heap(2*i +2) ) )
}

```