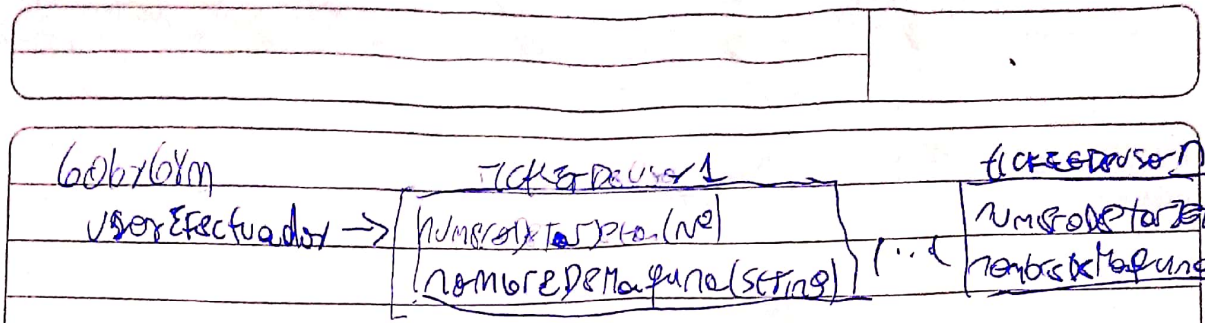

#TipoDeEjercicio -> MUSCULACION, CARDIO y FITNESS

Function CantidadDeAccesosAl_DeUsuarioEn(nombreMaquina, Gobierno){

/*

Proposito: DADO UN NOMBRE DE LA MAQUINA Y UN GOBIERNO, DESCRIBE LA CANTIDAD DE CONTRATACIONES QUE SE REALIZARON EN ESA MAQUINA

Precondicion: NINGUNA

Resultado: NUMERO

Parametros:

*nombreMaquina: String - ES EL NOMBRE DE LA MAQUINA EN EL CUAL SE DESEA SABER LA CANTIDAD DE CONTRATACIONES

*Gobierno: Gobierno: ES EL GOBIERNO EN EL CUAL SE DESEAN CONTAR LA CANTIDAD DE CONTRATACIONES QUE HUBO DE LA MAQUINA **nombreMaquina**

*/

CantidadDeContrataciones Victor. = 0

Foreach Usr In UsuarioEfectuador(Gobierno){

CantidadDeContrataciones Victor = CantidadDeContrataciones Victor +

UnosSi_SinoCero(nombreMaquina == Maquina)

}

Return (CantidadDeContrataciones Victor)

}

```
function maquinasQueSuperaronMonto_RegistradasEn(cantidad, gobsGym){
/*
  Proposito: Dado un numero *cantidad* y un sistema *gobsGym*,
             describe la lista de los nombres de las maquinas que se registraron
             mas tickets que la cantidad recibida
  Precondicion: Ninguna
  Devolucion: Lista de Strings
  Parametros:
        * cantidad - Numero - Es la cantidad de tickets a comparar
        * gobsGym - GobsGym; Es el sistema en el cual se desea
                    saber la lista de nombres de las maquinas que
                    superan los tickets de la cantidad dada
*/
nombresVistos := []
foreach uso in usosEfectuados(gobsGym){
    nombresVistos := nombresVistos ++ singularSi(nombreDeMaquina(uso),
                        cantidadDeAccesosAlRegistradosEn
                        (nombreDeMaquina(uso), gobsGym) > cantidad)
}
return (nombresVistos)
```

```
function sumarAccesosDe_A_En(nroTarjeta, listaDeNombresDeMaquinas, gobsGym){
/*
  Propósito: dado 1 número de tarjeta, una lista de nombres de máquinas y un
  gobsGym, describe el gobsGym resultante de intentar registrar el uso de
  la tarjeta dada por el número de tarjeta que se supone existente, en cada
  una de las máquinas cuyos nombres están en la lista dada.
  Precondición: el nroTarjeta existe y las máquinas no están en reparación
  Resultado: GobsGym
  Parámetros:
      * nroTarjeta: Número: es el Nº de tarjeta que se quiere registrar
        en el gobsGym dado
      * listaDeNombresDeMaquinas: Lista de Strings: es la lista de nombres de máquinas
        que se desean registrar al gobsGym con el Nº de tarjeta
        dado
      * gobsGym: es el sistema que se desea registrar los accesos.
*/
  return (GobsGym(gobsGym | usosEfectuados <- usosEfectuados(gobsGym) ++
                               usosDe_conLasMaquinas_(nroTarjeta,
                                  (listaDeNombresDe
                                   Maquinas)))
}
function usosDe_conLasMaquinas_(nroTarjeta, listaDeNombres){
/*
  Propósito: dado un nroTarjeta y una lista de nombres de máquinas describe
  una lista de usos efectuados con los usos de las máquinas de la lista dada
  usadas por el número de tarjeta.
  Precondición: nroTarjeta existe y las máquinas no están en reparación
  Resultado: lista de usos efectuados
  Parámetros:
      * nroTarjeta: es el número de tarjeta a registrar
      * listaDeNombres: es la lista de nombres de máquinas que se
        desea registrar su uso
*/
```

```
UserRegistrados := []
foreach maquina in listaDeNombres{
    UserRegistrados := UserRegistrados ++ [Maquina_UsadaPor_
                                           (Maquina, nroTarjeta)]
}
return (UserRegistrados)

}
Fnction Maquina_UsadaPor_ (nombreMaquina, nroTarjeta){
/*
Precondición: DADO UN NOMBRE DE MAQUINA Y UN NUMERO DE TARJETA REGISTRADA
              EL USO EFECTUADO DE LA MAQUINA CON NOMBRE DADO Y NRO
              DE TARJETA DADO.
Precondición: nroTarjeta EXISTE Y LA MAQUINA NO ESTA EN REPARACION
Resultado: TicketDeUso
Parámetros:
    * nombreMaquina: ES UN STRING QUE DESCRIBE EL
      NOMBRE DE LA MAQUINA A USAR
    * nroTarjeta: ES EL NRO DE LA TARJETA QUE SE QUIERE
                  REGISTRAR EL USO DE LA MAQUINA CON
                  NOMBRE DADO.
*/
return (TicketDeUso(numeroDeTarjeta -> nroTarjeta,
                    nombreDeMaquina -> nombreMaquina))
}
```

```
Function maquinasContratadasPor_En_(listaDeNumDeTalleres, sobreForm){
/*
Proposito: Para una lista de numeros de talleres y un sobreForm descarga una lista
de listas de maquinas.
Precondicion: Ninguna
Resultado: Lista de listas de maquinas.
Parametros:
  * listaDeNumDeTalleres: Lista de numeros - Esos numeros de
                          talleres en el cual se desean saber las maquinas
                          de uso con su usuario
  * sobreForm: sobreForm: es el sistema en el cual se desean filtrar
               la info de las maquinas usadas por los
               talleres con ese numero dado.
*/
  maquinasContratadasVectores:=[ ]

  foreach nroTallerEnLista de listaDeNumDeTalleres{
    maquinasContratadasVectores:= maquinasContratadasVectores ++
                                  [maquinasContratadasPorElUsuario
                                  En_(nroTaller, sobreForm)]
  }
  return (maquinasContratadasVectores)
}
```

Entero MaquinasContratadosPorElUsuario-En(numtarjeta, Gobsgym){

/*

Datos: Dado el numero de tarjeta y un GobsGym Describe una lista de maquinas en el cual fueron usadas por el usuario con numero de tarjeta dada.

Precondicion: Ninguna

Resultado: Lista de maquinas

Excepciones: no considerar Default

*/

MaquinasContratadosVistas := []

foreach maquina in Maquinas(Gobsgym){

MaquinasContratadosVistas := MaquinasContratadosVistas ++

sinRepetirSi(maquina, laMaquinaFueUsadaPor_(nombreDelaMaquina(.