```
#define function unsigned int
#define var unsigned int

/************************/
   Funciones básicas.
/************************/
function cero(){
    return 0;
}

function cero(var X1){
    return 0;
}

function S(var X1){
    X1++;
    return X1;
}

function U1_1(var X1){
    return X1;
}

function U2_1(var X1, var X2){
    return X1;
}

function U2_2(var X1, var X2){
    return X2;
}

function U3_1(var X1, var X2, var X3){
    return X1;
}
```

```
function U3_2(var X1, var X2, var X3){
    return X2;
}

function U3_3(var X1, var X2, var X3){
    return X3;
}

function U4_1(var X1, var X2, var X3, var X4){
    return X1;
}

function U4_2(var X1, var X2, var X3, var X4){
    return X2;
}

function U4_3(var X1, var X2, var X3, var X4){
    return X3;
}

function U4_4(var X1, var X2, var X3, var X4){
    return X4;
}

function U5_1(var X1, var X2, var X3, var X4, var X5){
    return X1;
}

function U5_2(var X1, var X2, var X3, var X4, var X5){
    return X2;
}

function U5_3(var X1, var X2, var X3, var X4, var X5){
    return X3;
}
```

```
76 function U5_4(var X1, var X2, var X3, var X4, var X5){
77     return X4;
78 }
79
80 function U5_5(var X1, var X2, var X3, var X4, var X5){
81     return X5;
82 }
83
84 /*********************************************/
85 /* Devuelve siempre 0. */
86 function cero(var X1, var X2){
87     var V1 = U2_1(X1, X2);
88     return  cero(V1);
89 }
90
91 /* Devuelve siempre 1. */
92 function uno(var X1){
93     var V1 = cero(X1);
94     return  S(V1);
95 }
96
97 /* Devuelve siempre 1. */
98 function uno(var X1, var X2, var X3){
99     var V1 = U3_2(X1, X2, X3);
100    return  uno(V1);
101 }
102
103 /* Devuelve siempre 2. */
104 function dos(var X1){
105    var V1 = uno(X1);
106    return  S(V1);
107 }
108
109 /* Devuelve siempre 2. */
110 function dos(var X1, var X2){
111    var V1 = U2_1(X1, X2);
112    return  dos(V1);
113 }
```

```
/* Devuelve siempre 2. */
function dos(var X1, var X2, var X3){
    var V1 = U3_2(X1, X2, X3);
    return  dos(V1);
}

/* Devuelve siempre 3. */
function tres(var X1, var X2){
    var V1 = dos(X1, X2);
    return  S(V1);
}

/* Devuelve siempre 3. */
function tres(var X1, var X2, var X3){
    var V1 = dos(X1, X2, X3);
    return  S(V1);
}

/* Devuelve siempre 4. */
function cuatro(var X1, var X2){
    var V1 = tres(X1, X2);
    return  S(V1);
}

/* Devuelve siempre 4. */
function cuatro(var X1, var X2, var X3){
    var V1 = tres(X1, X2, X3);
    return  S(V1);
}

/* ---------------------------------------------- */

/* Función auxiliar para la función «suma». */
function suma_aux(var X1, var X2, var X3){
    var V1 = U3_2(X1, X2, X3){
    return S(V1);
}
```

```
/* Devuelve la suma de X1 y X2. */
function suma(var X1, var X2){
    if(X1 == 0){
        return U1_1(X2);
    }
    else{
        X1--;
        var rr = suma(X1, X2);
        return suma_aux(X1, rr, X2);
    }
}

/* ---------------------------------------------- */

/* Auxiliar para «suma3». */
function suma_aux3(var X1, var X2, var X3){
    var V1 = U3_2(X1, X2, X3);
    var V2 = U3_3(X1, X2, X3);
    return suma(V1, V2);
}

/* Devuelve la suma de X1, X2 y X3. */
function suma3(var X1, var X2, var X3){
    var V1 = U3_1(X1, X2, X3);
    var V2 = suma_aux3(X1, X2, X3);
    return suma(V1, V2);
}

/* Auxiliar para «suma4». */
function suma_aux4(var X1, var X2, var X3, var X4){
    var V1 = U4_2(X1, X2, X3, X4);
    var V2 = U4_3(X1, X2, X3, X4);
    var V3 = U4_4(X1, X2, X3, X4);
    return suma3(V1, V2, V3);
}
```

```
/* Devuelve la suma de X1, X2, X3 y X4. */
function suma4(var X1, var X2, var X3, var X4){
    var V1 = U4_1(X1, X2, X3, X4);
    var V2 = suma_aux4(X1, X2, X3, X4);
    return suma(V1, V2);
}

/* Auxiliar para «suma5». */
function suma_aux5(var X1, var X2,var X3, var X4, var X5){
    var V1 = U5_2(X1, X2, X3, X4, X5);
    var V2 = U5_3(X1, X2, X3, X4, X5);
    var V3 = U5_4(X1, X2, X3, X4, X5);
    var V4 = U5_5(X1, X2, X3, X4, X5);
    return suma4(V1, V2, V3, V4);
}

/* Devuelve la suma de X1, X2, X3, X4 y X5. */
function suma5(var X1, var X2, var X3, var X4, var X5){
    var V1 = U5_1(X1, X2, X3, X4, X5);
    var V2 = suma_aux5(X1, X2, X3, X4, X5);
    return suma(V1, V2);
}

/* ---------------------------------------------- */

/* Función auxiliar para la función «resta».
 * devuelve X1 - 1. */
function anterior(var X1){
    if(X1 == 0){
        return cero();
    }

    else{
        X1--;
        var rr = anterior(X1);
        return U2_1(X1, rr);
    }
}

```

```
228 /* Función auxiliar para la función «resta». */
229 function resta_aux2(var X1, var X2, var X3){
230     var V1 = U3_2(X1, X2, X3);
231     return anterior(V1);
232 }
233
234 /* Función auxiliar para la función «resta». */
235 function resta_aux1(var X1, var X2){
236     if(X1 == 0){
237         return U1_1(X2);
238     }
239     else{
240         X1--;
241         var rr = resta_aux1(X1, X2);
242         return resta_aux2(X1, rr, X2);
243     }
244 }
245
246 /* Devuelve X1 menos X2. */
247 function resta(var X1, var X2){
248     var V1 = U2_2(X1, X2);
249     var V2 = U2_1(X1, X2);
250     return resta_aux1(V1, V2);
251 }
252
253 /* ---------------------------------------------- */
254
255 /* Función auxiliar para la función «mult». */
256 function mult_aux(var X1, var X2, var X3){
257     var V1 = U3_2(X1, X2, X3);
258     var V2 = U3_3(X1, X2, X3);
259     return suma(V1, V2);
260 }
261
262
263
264
265
```

```
266 /* Devuelve la multiplicación de X1 y X2. */
267 function mult(var X1, var X2){
268     if(X1 == 0){
269         return cero(X2);
270     }
271     else{
272         X1--;
273         var rr = mult(X1, X2);
274         return mult_aux(X1, rr, X2);
275     }
276 }
277
278 /* ---------------------------------------------- */
279
280 /* Función auxiliar para la función «mult_veces». */
281 function mult_veces_aux(var X1, var X2, var X3, var X4){
282     var V1 = U4_2(X1, X2, X3, X4);
283     var V2 = U4_3(X1, X2, X3, X4);
284     return mult(V1, V2);
285 }
286
287 /* Devuelve X3 multiplicado X1 veces por X2;
288  * esto es, X3 * (X2 ^ X1). */
289 function mult_veces(var X1, var X2, var X3){
290     if(X1 == 0){
291         return U2_2(X2, X3);
292     }
293     else{
294         X1--;
295         var rr = mult_veces(X1, X2, X3);
296         return mult_veces_aux(X1, rr, X2, X3);
297     }
298 }
299
300 /* ---------------------------------------------- */
301
302
303
```

```
/* Devuelve 0 si X1 es mayor que 0 y 1 si es 0. */
function no_sg(var X1){
    var V1 = uno(X1);
    var V2 = U1_1(X1);
    return resta(V1, V2);
}

/* Devuelve 0 si X1 es 0 y 1 si es mayor que 0. */
function sg(var X1){
    var V1 = uno(X1);
    var V2 = no_sg(X1);
    return resta(V1, V2);
}

/* Devuelve 1 si X1 es menor o igual que X2;
 * 0 en caso contrario. */
function menor_ig(var X1, var X2){
    var V1 = resta(X1, X2);
    return no_sg(V1);
}

/* Devuelve 1 si X1 es mayor o igual que X2;
 *  0 en caso contrario. */
function mayor_ig(var X1, var X2){
    var V1 = U2_2(X1, X2);
    var V2 = U2_1(X1, X2);
    return menor_ig(V1, V2);
}

/* Devuelve 1 si X1 es igual que X2; 0 en caso contrario. */
function igual(var X1, var X2){

    var V1 = mayor_ig(X1, X2);
    var V2 = menor_ig(X1, X2);
    return mult(V1, V2);
}

/* ---------------------------------------------- */

```

```
/* Declaración para funciones que usan recursión indirecta. */
function div_emp(var, var, var, var);

/* Función auxiliar para «div».
 * Comprueba si dividendo es mayor que el divisor. */
function div_aux_mayor_ig(var X1, var X2, var X3){
    var V1 = U3_2(X1, X2, X3);
    var V2 = U3_3(X1, X2, X3);
    return mayor_ig(V1, V2);
}

/* Función auxiliar para «div».
 * Suma uno al resultado provisional. */
function div_aux_sig(var X1, var X2,var X3,var X4,var X5){
    var V1 = U5_3(X1, X2, X3, X4, X5);
    return S(V1);
}

/* Función auxiliar para «div».
 * Resta divisor al dividendo. */
function div_aux_rt(var X1, var X2, var X3,var X4,var X5){
    var V1 = U5_4(X1, X2, X3, X4, X5);
    var V2 = U5_5(X1, X2, X3, X4, X5);
    return resta(V1, V2);
}

/* Función auxiliar para «div».
 * Suma uno al resultado, resta al
 * dividendo el divisor y empieza el ciclo. */
function div_aux2(var X1, var X2, var X3, var X4, var X5){
    var V1 = div_aux_sig(X1, X2, X3, X4, X5);

    var V3 = div_aux_rt(X1, X2, X3, X4, X5);
    var V2 = U5_5(X1, X2, X3, X4, X5);
    return div_emp(V1, V2, V3);
}
```

```
/* Función auxiliar para «div».
 * X1: ¿es dividendo mayor que divisor?
 * X2: resultado provisional.
 * X3: dividendo.
 * X4: divisor. */
function div_aux1(var X1, var X2, var X3, var X4){
    if(X1 == 0){
        return U3_1(X2, X3, X4);
    }
    else{
        X1--;
        var rr = div_aux1(X1, X2, X3, X4);
        return div_aux2(X1, rr, X2, X3, X4);
    }
}

/* Función auxiliar para «div».
 * X1: resultado provisional.
 * X2: dividendo.
 * X3: divisor. */
function div_emp(var X1, var X2, var X3){
    var V1 = div_aux_mayor_ig(X1, X2, X3);
    var V2 = U3_1(X1, X2, X3);
    var V3 = U3_2(X1, X2, X3);
    var V4 = U3_3(X1, X2, X3);
    return div_aux1(V1, V2, V3, V4);
}

/* Devuelve n, donde n = X1/X2.
 * X1: dividendo.
 * X2: divisor. */
function div(var X1, var X2){
    var V1 = cero(X1, X2);
    var V2 = U2_1(X1, X2);
    var V3 = U2_2(X1, X2);
    return div_emp(V1, V2, V3);
}

```

```
418 /* ---------------------------------------------- */
419
420 /* Función auxiliar para «resto».
421  * Devuelve multiplicación de división y divisor. */
422 function resto_aux2(var X1, var X2, var X3){
423     var V1 = U3_1(X1, X2, X3);
424     var V2 = U3_3(X1, X2, X3);
425     return mult(V1, V2);
426 }
427
428 /* Función auxiliar para «resto». Resta al
429  * dividendo la multiplicación de división y divisor. */
430 function resto_aux1(var X1, var X2, var X3){
431     var V1 = U3_2(X1, X2, X3);
432     var V2 = resto_aux2(X1, X2, X3);
433     return resta(V1, V2);
434 }
435
436 /* Devuelve el resto de X1/X2, esto es, X1 mod X2.
437  * X1: dividendo.
438  * X2: divisor. */
439 function resto(var X1, var X2){
440     var V1 = div(X1, X2);
441     var V2 = U2_1(X1, X2);
442     var V3 = U2_2(X1, X2);
443     return resto_aux1(V1, V2, V3);
444 }
445
446 /* ---------------------------------------------- */
447
448 /* Devuelve 1 si X1 es divisible entre X2;
449  * 0 en caso contrario. */
450 function es_div(var X1, var X2){
451     var V1 = resto(X1, X2);
452     return no_sg(V1);
453 }
454
455
```

```
/* Devuelve 1 si X1 no es divisible
 * entre X2; 0 en caso contrario. */
function no_es_div(var X1, var X2){
    var V1 = es_div(X1, X2);
    return no_sg(V1);
}

/* ---------------------------------------------- */

/* Declaración para funciones que usan recursión indirecta. */
function vdiv_emp(var, var, var);

/* Función auxiliar para «vdiv».
 * Devuelve 1 si es divisible X2 entre X3; si no 0. */
function vdiv_aux_es_div(var X1, var X2, var X3){
    var V1 = U3_2(X1, X2, X3);
    var V1 = U3_3(X1, X2, X3);
    return es_div(V1, V2);
}

/* Función auxiliar para «vdiv».
 * Divide X4 entre X5. */
function vdiv_aux_div(var X1,var X2,var X3,var X4,var X5){
    var V1 = U5_4(X1, X2, X3, X4, X5);
    var V2 = U5_5(X1, X2, X3, X4, X5);
    return div(V1, V2);
}

/* Función auxiliar para «vdiv».
 * Suma 1 al resultado provisional y divide
 * dividendo entre divisor. */
function vdiv_aux2(var X1, var X2, var X3, var X4,var X5){
    var V1 = div_aux_sig(X1, X2, X3, X4, X5);
    var V2 = vdiv_aux_div(X1, X2, X3, X4, X5);
    var V3 = U5_5(X1, X2, X3, X4, X5);
    return vdiv_emp(V1, V2, V2);
}

```

```
/* Función auxiliar para «vdiv». Devuelve el resultado
 * si X3 no es divisible entre X4; si no, sigue el ciclo. */
function vdiv_aux1(var X1, var X2, var X3, var X4){
    if(X1 == 0){
        return U3_1(X2, X3, X4);
    }
    else{
        X1--;
        var rr = vdiv_aux1(X1, X2, X3, X4);
        return vdiv_aux2(X1, rr, X2, X3, X4);
    }
}

/* Función auxiliar para «vdiv». Comienza el ciclo. */
function vdiv_emp(var X1, var X2, var X3){
    var V1 = vdiv_aux_es_div(X1, X2, X3);
    var V2 = U3_1(X1, X2, X3);
    var V3 = U3_2(X1, X2, X3);
    var V4 = U3_3(X1, X2, X3);
    return vdiv_aux1(V1, V2, V3, V4);
}

/* Devuelve las veces que es divisible X1 entre X2. */
function vdiv(var X1, var X2){
    var V1 = cero(X1, X2);
    var V2 = U2_1(X1, X2);
    var V3 = U2_2(X1, X2);
    return vdiv_emp(V1, V2, V3);
}

/* ---------------------------------------------- */

/* Función auxiliar para «nd_hasta_aux2».
 * Suma 2 para que no se compruebe si es divisible entre 0 o 1.*/
function nd_hasta_aux3(var X1, var X2, var X3){
    var V1 = U3_1(X1, X2, X3);
    var V2 = dos(X1, X2, X3);
    return suma(V1, V2);
}
```

```
/* Función auxiliar para «nd_hasta_aux1». */
function nd_hasta_aux2(var X1, var X2, var X3){
    var V1 = U3_3(X1, X2, X3);
    var V2 = nd_hasta_aux3(X1, X2, X3);
    return no_es_div(V1, V2);
}

/* Función auxiliar para «no_div_hasta». */
function nd_hasta_aux1(var X1, var X2, var X3){
    var V2 = nd_hasta_aux2(X1, X2, X3);
    var V1 = U3_2(X1, X2, X3);
    return mult(V1, V2);
}

/* Función auxiliar para «es_primo».
 * Devuelve 1 si no hay divisor de X2 entre 2 y X1 + 1. */
function no_div_hasta(var X1, var X2){
    if(X1 == 0){
        return uno(X2);
    }
    else{
        X1--;
        var rr = no_div_hasta(X1, X2);
        return nd_hasta_aux1(X1, rr, X2);
    }
}

/* Función auxiliar para «es_primo». */
function partir(var X1){
    var V1 = anterior(X1);
    var V2 = dos(X1);

    return div(V1, V2);
}
```

```
/* Devuelve 1 si X1 es primo; 0 en caso contrario. */
function es_primo(var X1){
    var V1 = partir(X1);
    var V2 = U1_1(X1);
    return no_div_hasta(V1, V2);
}

/* Devuelve 0 si X1 es primo; 1 en caso contrario. */
function no_es_primo(var X1){
    var V1 = es_primo(X1);
    return no_sg(V1);
}

/* ------------------------------------------------ */

/* Declaración de función. */
function sig_primo_desde(var);

/* Función auxiliar para «sig_primo_aux1». */
function sig_primo_aux2(var X1, var X2, var X3){
    var V1 = U3_2(X1, X2, X3);
    return sig_primo_desde(V1);
}

/* Si X2 es primo lo devuelve; si no
 * sigue buscando el menor primo mayor que X2. */
function sig_primo_aux1(var X1, var X2){
    if(X1 == 0){
        return U1_1(X2);
    }
    else{

        X1--;
        var rr = sig_primo_aux1(X1, X2);
        return sig_primo_aux2(X1, rr, X2);
    }
}


```

```
608 /* Devuelve el menor primo mayor o igual que X1. */
609 function sig_primo(var X1){
610     var V1 = no_es_primo(X1);
611     var V2 = U1_1(X1);
612     return sig_primo_aux1(V1, V2);
613 }
614
615 /* Devuelve el menor primo mayor que X1. */
616 function sig_primo_desde(var X1){
617     var V1 = S(X1);
618     return sig_primo(V1);
619 }
620
621 /* Devuelve el menor primo mayor que el primo X2. */
622 function inst_primo_aux(var X1, var X2){
623     var V1 = U2_2(X1, X2);
624     return sig_primo_desde(V1);
625 }
626
627 /* Devuelve el primo con el que se
628  * codifica la instrucción en la posición X1. */
629 function inst_primo(var X1){
630     if(X1 == 0){
631         return dos();
632     }
633     else{
634         X1--;
635         var rr = inst_primo(X1);
636         return inst_primo_aux(X1, rr);
637     }
638 }

639 /* ---------------------------------------------- */
640 /* Devuelve el primo correspondiente a la posición X1.
641  * X1: posición. */
642 function primo_num_inst(var X1, var X2){
643     var V1 = U2_1(X1, X2);
644     return inst_primo(V1);
645 }
```

```
/* Devuelve la instrucción en la posición X1.
 * X1: posición.
 * X2: registro. */
function inst(var X1, var X2){
    var V1 = U2_2(X1, X2);
    var V2 = primo_num_inst(X1, X2);
    return vdiv(V1, V2);
}

/* Devuelve 1 si X1 es instrucción; 0 en caso contrario.
 * X1: instrucción. */
function hay_inst(var X1, var X2){
    var V1 = U2_1(X1, X2);
    return sg(V1);
}

/* Devuelve: 0 si no hay instrucción o ésta es '1';
 * 1 si '0'.
 * 2 si '='.
 * 3 si '*'. */
function tipo_inst_aux(var X1, var X2){
    var V1 = U2_1(X1, X2);
    var V2 = cuatro(X1, X2);
    return resto(V1, V2);
}

/* Devuelve, según la instrucción en X1:
 * 0 si no hay instrucción.
 * 1 si '1'.
 * 2 si '0'.
 * 3 si '='.
 * 4 si '*'. */
function tipo_inst(var X1, var X2){
    var V1 = tipo_inst_aux(X1, X2);
    var V2 = hay_inst(X1, X2);
    return suma(V1, V2);
}
```

```
684 /* Auxiliar para «pos_inst».
685  * suma 3 para que al dividir entre 4
686  * no dé nunca 0 y dé el resultado correcto. */
687 function pos_inst_aux(var X1, var X2){
688     var V1 = U2_1(X1, X2);
689     var V2 = tres(X1, X2);
690     return suma(V1, V2);
691 }
692
693 /* Devuelva la posición a la
694  * que se refiere la instrucción X1. */
695 function pos_inst(var X1, var X2){
696     var V1 = pos_inst_aux(X1, X2);
697     var V2 = cuatro(X1, X2);
698     return div(V1, V2);
699 }
700
701 /* Devuelva la posición a la que
702   * señala el puntero en el registro X1. */
703 function sacar_puntero(var X1){
704     var V1 = U1_1(X1);
705     var V2 = dos(X1);
706     return vdiv(V1, V2);
707 }
708
709 /* Devuelve el registro con
710  * el puntero avanzado una posición. */
711 function avanzar_inst(var X1, var X2){
712     var V1 = dos(X1, X2);
713     var V2 = U2_2(X1, X2);
714     return mult(V1, V2);
715 }
716
717 /* ---------------------------------------------- */
718
719
720
721
```

```
/* Devuelve el registro X2 modificado con una marca más
 * en la posición X1, o un símbolo «1» si estaba vacía,
 * y el puntero señalando a la siguiente posición. */
function marcar(var X1, var X2){
    var V1 = cuatro(X1, X2);
    var V2 = primo_num_inst(X1, X2);
    var V3 = avanzar_inst(X1, X2);
    return mult_veces(V1, V2, V3);
}
/* ---------------------------------------------- */

/* Auxiliar para «borrar_aux2». Resta para mantener
 * el símbolo en la posición referida si éste no es '1'. */
function borrar_aux3(var X1, var X2){
    var V1 = U2_1(X1, X2);
    var V2 = tipo_inst_aux(X1, X2);
    return resta(V1, V2);
}

/* Auxiliar para «borrar_aux».
 * X1: instrucción en posición referida.
 * X2: primo en posición referida. */
function borrar_aux2(var X1, var X2){
    var V1 = borrar_aux3(X1, X2);
    var V2 = U2_2(X1, X2);
    var V2 = uno(X1, X2);
    return mult_veces(V1, V2, V3);
}

/* Devuelve el número entre el que hay que dividir
 * el registro X2 para borrar la posición X1.

 * X1: posición referida.
 * X2: registro. */
function borrar_aux1(var X1, var X2){
    var V1 = inst(X1, X2);
    var V2 = primo_num_inst(X1, X2);
    return borrar_aux2(V1, V2);
}
```

```
/* Devuelve el registo con la posición X1 borrada.
 * X1: posición referida.
 * X2: registro. */
function borrar(var X1, var X2){
    var V1 = avanzar_inst(X1, X2);
    var V2 = borrar_aux1(X1, X2);
    return div(V1, V2);
}

/* ---------------------------------------------- */


/* Devuelve la posición del puntero en registro X2. */
function quitar_puntero_aux2(var X1, var X2){
    var V1 = U2_2(X1, X2);
    return sacar_puntero(V1);
}

/* Devuelve el número entre el que hay
 *  que dividir registro X2 para quitarle el puntero. */
function quitar_puntero_aux1(var X1, var X2){
    var V1 = quitar_puntero_aux2(X1, X2);
    var V2 = dos(X1, X2);
    var V3 = uno(X1, X2);
    return mult_veces(V1, V2, V3);
}

/* Devuelve el registro X2 sin puntero. */
function quitar_puntero(var X1, var X2){
    var V1 = quitar_puntero_aux1(X1, X2);
    var V2 = U2_2(X1, X2);

    return div(V1, V2);
}
```

```
798 /* Devuelve el registro X2 con el
799  * puntero señalando a la posición X1. */
800 function saltar(var X1, var X2){
801     var V1 = U2_1(X1, X2);
802     var V2 = dos(X1, X2);
803     var V3 = quitar_puntero(X1, X2);
804     return mult_veces(V1, V2, V3);
805 }
806
807 /* ---------------------------------------------- */
808
809
810 /* Devuelve la instrucción
811  * en la primera posición de registro X2. */
812 function inst_uno(var X1, var X2){
813     var V1 = uno(X1, X2);
814     var V2 = U2_2(X1, X2);
815     return inst(V1, V2);
816 }
817
818 /* Devuelve la posición referida por la
819  * instrucción en la primera posición de registro X2. */
820 function v_pos_uno(var X1, var X2){
821     var V1 = inst_uno(X1, X2);
822     var V2 = U2_2(X1, X2);
823     return pos_inst(V1, V2);
824 }
825
826 /* Devuelve la posición referida por la
827  * instrucción X1 de registro X2. */
828 function v_pos(var X1, var X2){
829     var V1 = inst(X1, X2);
830     var V2 = U2_2(X1, X2);
831     return pos_inst(V1, V2);
832 }
833
834
835
```

```
836 /* Devuelve 1 si la posición X1 de registro X2
837  * es igual que la primera; 0 en caso contrario. */
838 function hay_salto(var X1, var X2){
839     var V1 = v_pos_uno(X1, X2);
840     var V2 = v_pos(X1, X2);
841     return es_igual(V1, V2);
842 }
843
844 /* Devuelve el registro X2 con el puntero
845  * señalando a la siguiente posición de la actual si
846  * la posición X1 es distinta de la primera posición;
847  * a la siguiente de la siguiente en caso contrario. */
848 function comparar(var X1, var X2){
849     var V1 = hay_salto(X1, X2);
850     var V2 = dos(X1, X2);
851     var V3 = avanzar_inst(X1, X2);
852     return mult_veces(V1, V2, V3);
853 }
854
855 /* ---------------------------------------------- */
856
857
858 /* Devuelve el registro X3 después de ejecutar 'marcar'. */
859 function eval_marcar_aux(var X1, var X2, var X3){
860     var V1 = U3_2(X1, X2, X3);
861     var V2 = U3_3(X1, X2, X3);
862     return marcar(V1, V2);
863 }
864
865 /* Si el tipo de instrucción X1 es 'marcar',
866  * devuelve 1; 0 en caso contrario. */
867 function es_marcar(var X1, var X2, var X3){
868     var V1 = U3_1(X1, X2, X3);
869     var V2 = uno(X1, X2, X3);
870     return es_igual(V1, V2);
871 }
872
873
```

```
/* Si X1 es 1 devuelve el registro X3
 * después de ejecutar la instrucción 'marcar'
 * en la posición X2; devuelve 0 si X1 no es 1. */
function eval_marcar(var X1, var X2, var X3){
    var V1 = es_marcar(X1, X2, X3);
    var V2 = eval_marcar_aux(X1, X2, X3);
    return mult(V1, V2);
}

/* ---------------------------------------------- */

/* Devuelve el registro X3 después de ejecutar 'borrar'. */
function eval_borrar_aux(var X1, var X2, var X3){
    var V1 = U3_2(X1, X2, X3);
    var V2 = U3_3(X1, X2, X3);
    return borrar(V1, V2);
}

/* Si el tipo de instrucción X1 es
 * 'borrar', devuelve 1; 0 en caso contrario. */
function es_borrar(var X1, var X2, var X3){
    var V1 = U3_1(X1, X2, X3);
    var V2 = dos(X1, X2, X3);
    return es_igual(V1, V2);
}

/* Si X1 es 2 devuelve el registro X3
 * después de ejecutar la instrucción 'borrar'
 * en la posición X2; devuelve 0 si X1 no es 2. */
function eval_borrar(var X1, var X2, var X3){
    var V1 = es_borrar(X1, X2, X3);
    var V2 = eval_borrar_aux(X1, X2, X3);
    return mult(V1, V2);
}

/* ---------------------------------------------- */
```

```
/* Devuelve el registro X3 después de ejecutar 'comparar'. */
function eval_comparar_aux(var X1, var X2, var X3){
    var V1 = U3_2(X1, X2, X3);
    var V2 = U3_3(X1, X2, X3);
    return comparar(V1, V2);
}

/* Si el tipo de instrucción X1 es
 * 'comparar', devuelve 1; 0 en caso contrario. */
function es_comparar(var X1, var X2, var X3){
    var V1 = U3_1(X1, X2, X3);
    var V2 = tres(X1, X2, X3);
    return es_igual(V1, V2);
}

/* Si X1 es 3 devuelve el registro X3
 * después de ejecutar la instrucción 'comparar'
 * en la posición X2; devuelve 0 si X1 no es 3. */
function eval_comparar(var X1, var X2, var X3){
    var V1 = es_comparar(X1, X2, X3);
    var V2 = eval_comparar_aux(X1, X2, X3);
    return mult(V1, V2);
}
/* ------------------------------------------------ */
/* Devuelve el registro X3 después de ejecutar 'saltar'. */
function eval_saltar_aux(var X1, var X2, var X3){
    var V1 = U3_2(X1, X2, X3);
    var V2 = U3_3(X1, X2, X3);
    return saltar(V1, V2);
}

/* Si el tipo de instrucción X1 es 'saltar', devuelve 1;
 * 0 en caso contrario. */
function es_saltar(var X1, var X2, var X3){
    var V1 = U3_1(X1, X2, X3);
    var V2 = cuatro(X1, X2, X3);
    return es_igual(V1, V2);
}
```

```
/* Si X1 es 4 devuelve el registro X3
 * después de ejecutar la instrucción 'saltar'
 * en la posición X2; devuelve 0 si X1 no es 4. */
function eval_saltar(var X1, var X2, var X3){
    var V1 = es_saltar(X1, X2, X3);
    var V2 = eval_saltar_aux(X1, X2, X3);
    return mult(V1, V2);
}


/* ---------------------------------------------- */


/* Función auxiliar para «eval_no_inst». */
function eval_no_inst_aux(var X1, var X2, var X3){
    var V1 = U3_1(X1, X2, X3);
    return no_sg(V1);
}

/* Devuelve el registro X3 sin modificar
 * si el tipo de instrucción X1 es 0;
 * devuelve 0 en caso contrario. */
function eval_no_inst(var X1, var X2, var X3){
    var V1 = no_inst_aux(X1, X2, X3);
    var V2 = U3_3(X1, X2, X3);
    return mult(V1, V2);
}


/* ---------------------------------------------- */


/* Devuelve 1 si hay instrucción en X1; 0 en caso contrario . */
function hay_inst(var X1, var X2, var X3){
    var V1 = U3_1(X1, X2, X3);
    return sg(V1);
}

```

```
988 /* Devuelve el registro después de ejecutar la intrucción
989  * que indica X1 o el registro sin tocar si X1 es 0. */
990 function eval_registro(var X1, var X2, var X3){
991     var V1 = eval_marcar(X1, X2, X3);
992     var V2 = eval_borrar(X1, X2, X3);
993     var V3 = eval_comparar(X1, X2, X3);
994     var V4 = eval_saltar(X1, X2, X3);
995     var V5 = eval_no_inst(X1, X2, X3);
996     return suma(V1, V2, V3, V4, V5);
997 }
998
999 /* Comprueba si hay instrucción, ejecuta
1000  * en caso de haberla y vuelve a empezar el ciclo. */
1001 function ejec3(var X1, var X2, var X3){
1002     var V1 = hay_inst(X1, X2, X3);
1003     var V2 = eval_registro(X1, X2, X3);
1004     return evaluar(V1, V2);
1005 }
1006
1007 /* Saca el tipo de instrucción, la
1008  * posición referida y llama a «ejec3».
1009  * X1: instrucción.
1010  * X2: registro. */
1011 function ejec2(var X1, var X2){
1012     var V1 = tipo_inst(X1, X2);
1013     var V2 = pos_inst(X1, X2);
1014     var V3 = U2_2(X1, X2);
1015     return ejec3(V1, V2, V3);
1016 }
1017
1018 /* Saca la instrucción y llama a «ejec2» con ésta y el registro X2.
1019  * X1: puntero.
1020  * X2: registro. */
1021 function ejec1(var X1, var X2){
1022     var V1 = inst(X1, X2);
1023     var V2 = U2_2(X1, X2);
1024     return ejec2(V1, V2);
1025 }
```

```
/* Toma el registro X1 y llama a «ejec1»
 * con éste y el puntero contenido en él. */
function ejecutar_inst(var X1){
    var V1 = sacar_puntero(X1);
    var V2 = U1_1(X1);
    return ejec1(V1, V2);
}

/* X3: registro. */
function ejecutar(var X1, var X2, var X3){
    var V1 = U3_3(X1, X2, X3);
    return ejecutar_inst(V1);
}

/*
 * Si X1 es 0 devuelve el registro y acaba la computación;
 * si es 1 ejecuta la siguiente instrucción, si la hubiera,
 * en registro X2; es decir, ejecuta el siguienet paso.
 */
function evaluar(var X1, var X2){
    if(X1 == 0){
        return U1_1(X2);
    }
    else{
        X1--;
        var rr = evaluar(X1, X2);
        return ejecutar(X1, rr, X2);
    }
}

/* Auxiliar para «computar». */
function computar_aux(var X1, var X2){
    var V1 = uno(X1, X2);
    var V2 = mult(X1, X2);
    return evaluar(V1, V2);
}
```

```
/*
 * Función recursiva primitiva que computa el mismo valor
 * que el programa C-- que recibe como argumento; en caso de termina
 * devuelve un registro con el puntero y programa después del cómput
 */
function computar(var X1){
    var V1 = dos(X1);
    var V2 = U1_1(X1);
    return computar_aux(V1, V2);
}
```