```
#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;
}

function U5_4(var X1, var X2, var X3, var X4, var X5){
    return X4;
}

function U5_5(var X1, var X2, var X3, var X4, var X5){
    return X5;
}
```

```
78 /***********************************************/
79
80 /* Devuelve siempre 0. */
81 function cero(var X1, var X2){
82     var V1 = U2_1(X1, X2);
83     return  cero(V1);
84 }
85
86 /* Devuelve siempre 1. */
87 function uno(var X1){
88     var V1 = cero(X1);
89     return  S(V1);
90 }
91
92 /* Devuelve siempre 1. */
93 function uno(var X1, var X2, var X3){
94     var V1 = U3_2(X1, X2, X3);
95     return  uno(V1);
96 }
97
98 /* Devuelve siempre 2. */
99 function dos(var X1){
100     var V1 = uno(X1);
101     return  S(V1);
102 }
103
104 /* Devuelve siempre 2. */
105 function dos(var X1, var X2){
106     var V1 = U2_1(X1, X2);
107     return  dos(V1);
108 }
109
110 /* Devuelve siempre 2. */
111 function dos(var X1, var X2, var X3){
112     var V1 = U3_2(X1, X2, X3);
113     return  dos(V1);
114 }
```

```
/* 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);
}
```

```
146 /* Devuelve la suma de X1 y X2. */
147 function suma(var X1, var X2){
148     if(X1 == 0){
149         return U1_1(X2);
150     }
151     else{
152         X1--;
153         var rr = suma(X1, X2);
154         return suma_aux(X1, rr, X2);
155     }
156 }
157
158 /* ------------------------------------------------ */
159
160 /* Auxiliar para «suma3». */
161 function suma_aux3(var X1, var X2, var X3){
162     var V1 = U3_2(X1, X2, X3);
163     var V2 = U3_3(X1, X2, X3);
164     return suma(V1, V2);
165 }
166
167 /* Devuelve la suma de X1, X2 y X3. */
168 function suma3(var X1, var X2, var X3){
169     var V1 = U3_1(X1, X2, X3);
170     var V2 = suma_aux3(X1, X2, X3);
171     return suma(V1, V2);
172 }
173
174 /* Auxiliar para «suma4». */
175 function suma_aux4(var X1, var X2, var X3, var X4){
176     var V1 = U4_2(X1, X2, X3, X4);
177     var V2 = U4_3(X1, X2, X3, X4);
178     var V3 = U4_4(X1, X2, X3, X4);
179     return suma3(V1, V2, V3);
180 }
```

```
/* 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);
    }
}
```

```
/* Función auxiliar para la función «resta». */
function resta_aux2(var X1, var X2, var X3){
    var V1 = U3_2(X1, X2, X3);
    return anterior(V1);
}

/* Función auxiliar para la función «resta». */
function resta_aux1(var X1, var X2){
    if(X1 == 0){
        return U1_1(X2);
    }
    else{
        X1--;
        var rr = resta_aux1(X1, X2);
        return resta_aux2(X1, rr, X2);
    }
}

/* Devuelve X1 menos X2. */
function resta(var X1, var X2){
    var V1 = U2_2(X1, X2);
    var V2 = U2_1(X1, X2);
    return resta_aux1(V1, V2);
}

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

/* Función auxiliar para la función «mult». */
function mult_aux(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 multiplicación de X1 y X2. */
function mult(var X1, var X2){
    if(X1 == 0){
        return cero(X2);
    }
    else{
        X1--;
        var rr = mult(X1, X2);
        return mult_aux(X1, rr, X2);
    }
}

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

/* Función auxiliar para la función «mult_veces». */
function mult_veces_aux(var X1, var X2, var X3, var X4){
    var V1 = U4_2(X1, X2, X3, X4);
    var V2 = U4_3(X1, X2, X3, X4);
    return mult(V1, V2);
}

/* Devuelve X3 multiplicado X1 veces por X2;
 * esto es, X3 * (X2 ^ X1). */
function mult_veces(var X1, var X2, var X3){
    if(X1 == 0){
        return U2_2(X2, X3);
    }
    else{
        X1--;
        var rr = mult_veces(X1, X2, X3);
        return mult_veces_aux(X1, rr, X2, X3);
    }
}

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

```
/* 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);
}

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

```
/* Función auxiliar para «resto».
 * Devuelve multiplicación de división y divisor. */
function resto_aux2(var X1, var X2, var X3){
    var V1 = U3_1(X1, X2, X3);
    var V2 = U3_3(X1, X2, X3);
    return mult(V1, V2);
}

/* Función auxiliar para «resto». Resta al
 * dividendo la multiplicación de división y divisor. */
function resto_aux1(var X1, var X2, var X3){
    var V1 = U3_2(X1, X2, X3);
    var V2 = resto_aux2(X1, X2, X3);
    return resta(V1, V2);
}

/* Devuelve el resto de X1/X2, esto es, X1 mod X2.
 * X1: dividendo.
 * X2: divisor. */
function resto(var X1, var X2){
    var V1 = div(X1, X2);
    var V2 = U2_1(X1, X2);
    var V3 = U2_2(X1, X2);
    return resto_aux1(V1, V2, V3);
}

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

/* Devuelve 1 si X1 es divisible entre X2;
 * 0 en caso contrario. */
function es_div(var X1, var X2){
    var V1 = resto(X1, X2);

    return no_sg(V1);
}

```

```
435 /* Devuelve 1 si X1 no es divisible
436  * entre X2; 0 en caso contrario. */
437 function no_es_div(var X1, var X2){
438     var V1 = es_div(X1, X2);
439     return no_sg(V1);
440 }
441
442 /* ------------------------------------------------ */
443
444 /* Declaración para funciones que usan recursión indirecta. */
445 function vdiv_emp(var, var, var);
446
447 /* Función auxiliar para «vdiv».
448  * Devuelve 1 si es divisible X2 entre X3; si no 0. */
449 function vdiv_aux_es_div(var X1, var X2, var X3){
450     var V1 = U3_2(X1, X2, X3);
451     var V1 = U3_3(X1, X2, X3);
452     return es_div(V1, V2);
453 }
454
455 /* Función auxiliar para «vdiv».
456  * Divide X4 entre X5. */
457 function vdiv_aux_div(var X1,var X2,var X3,var X4,var X5){
458     var V1 = U5_4(X1, X2, X3, X4, X5);
459     var V2 = U5_5(X1, X2, X3, X4, X5);
460     return div(V1, V2);
461 }
462
463 /* Función auxiliar para «vdiv».
464  * Suma 1 al resultado provisional y divide
465  * dividendo entre divisor. */
466 function vdiv_aux2(var X1, var X2, var X3, var X4,var X5){
467     var V1 = div_aux_sig(X1, X2, X3, X4, X5);
468     var V2 = vdiv_aux_div(X1, X2, X3, X4, X5);
469     var V3 = U5_5(X1, X2, X3, X4, X5);
470     return vdiv_emp(V1, V2, V2);
471 }
472
```

```
473 /* Función auxiliar para «vdiv». Devuelve el resultado
474  * si X3 no es divisible entre X4; si no, sigue el ciclo. */
475 function vdiv_aux1(var X1, var X2, var X3, var X4){
476     if(X1 == 0){
477         return U3_1(X2, X3, X4);
478     }
479     else{
480         X1--;
481         var rr = vdiv_aux1(X1, X2, X3, X4);
482         return vdiv_aux2(X1, rr, X2, X3, X4);
483     }
484 }
485
486 /* Función auxiliar para «vdiv». Comienza el ciclo. */
487 function vdiv_emp(var X1, var X2, var X3){
488     var V1 = vdiv_aux_es_div(X1, X2, X3);
489     var V2 = U3_1(X1, X2, X3);
490     var V3 = U3_2(X1, X2, X3);
491     var V4 = U3_3(X1, X2, X3);
492     return vdiv_aux1(V1, V2, V3, V4);
493 }
494
495 /* Devuelve las veces que es divisible X1 entre X2. */
496 function vdiv(var X1, var X2){
497     var V1 = cero(X1, X2);
498     var V2 = U2_1(X1, X2);
499     var V3 = U2_2(X1, X2);
500     return vdiv_emp(V1, V2, V3);
501 }
502
503 /* ---------------------------------------------- */
504
505 /* Función auxiliar para «nd_hasta_aux2».
506  * Suma 2 para que no se compruebe si es divisible entre 0 o 1.*/
507 function nd_hasta_aux3(var X1, var X2, var X3){
508     var V1 = U3_1(X1, X2, X3);
509     var V2 = dos(X1, X2, X3);
510     return suma(V1, V2);
511 }
```

```
/* 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);
}
```

```
552 /* Devuelve 0 si X1 es primo; 1 en caso contrario. */
553 function no_es_primo(var X1){
554     var V1 = es_primo(X1);
555     return no_sg(V1);
556 }
557
558 /* ---------------------------------------------- */
559
560 /* Declaración de función. */
561 function sig_primo_desde(var);
562
563 /* Función auxiliar para «sig_primo_aux1». */
564 function sig_primo_aux2(var X1, var X2, var X3){
565     var V1 = U3_2(X1, X2, X3);
566     return sig_primo_desde(V1);
567 }
568
569 /* Si X2 es primo lo devuelve; si no
570  * sigue buscando el menor primo mayor que X2. */
571 function sig_primo_aux1(var X1, var X2){
572     if(X1 == 0){
573         return U1_1(X2);
574     }
575     else{
576         X1--;
577         var rr = sig_primo_aux1(X1, X2);
578         return sig_primo_aux2(X1, rr, X2);
579     }
580 }
581
582 /* Devuelve el menor primo mayor o igual que X1. */
583 function sig_primo(var X1){
584     var V1 = no_es_primo(X1);
585     var V2 = U1_1(X1);
586     return sig_primo_aux1(V1, V2);
587 }
588
```

```
589 /* Devuelve el menor primo mayor que X1. */
590 function sig_primo_desde(var X1){
591     var V1 = S(X1);
592     return sig_primo(V1);
593 }
594
595 /* Devuelve el menor primo mayor que el primo X2. */
596 function inst_primo_aux(var X1, var X2){
597     var V1 = U2_2(X1, X2);
598     return sig_primo_desde(V1);
599 }
600
601 /* Devuelve el primo con el que se
602  * codifica la instrucción en la posición X1. */
603 function inst_primo(var X1){
604     if(X1 == 0){
605         return dos();
606     }
607     else{
608         X1--;
609         var rr = inst_primo(X1);
610         return inst_primo_aux(X1, rr);
611     }
612 }
613 /* ------------------------------------------------ */
614 /* Devuelve el primo correspondiente a la posición X1.
615  * X1: posición. */
616 function primo_num_inst(var X1, var X2){
617     var V1 = U2_1(X1, X2);
618     return inst_primo(V1);
619 }
620 /* Devuelve la instrucción en la posición X1.
621  * X1: posición.
622  * X2: registro. */
623 function inst(var X1, var X2){
624     var V1 = U2_2(X1, X2);
625     var V2 = primo_num_inst(X1, X2);
626     return vdiv(V1, V2);
627 }
```

```
/* 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);
}

/* Auxiliar para «pos_inst».
 * suma 3 para que al dividir entre 4
 * no dé nunca 0 y dé el resultado correcto. */
function pos_inst_aux(var X1, var X2){
    var V1 = U2_1(X1, X2);
    var V2 = tres(X1, X2);
    return suma(V1, V2);
}

```

```
/* Devuelva la posición a la
 * que se refiere la instrucción X1. */
function pos_inst(var X1, var X2){
    var V1 = pos_inst_aux(X1, X2);
    var V2 = cuatro(X1, X2);
    return div(V1, V2);
}

/* Devuelva la posición a la que
  * señala el puntero en el registro X1. */
function sacar_puntero(var X1){
    var V1 = U1_1(X1);
    var V2 = dos(X1);
    return vdiv(V1, V2);
}

/* Devuelve el registro con
 * el puntero avanzado una posición. */
function avanzar_inst(var X1, var X2){
    var V1 = dos(X1, X2);
    var V2 = U2_2(X1, X2);
    return mult(V1, V2);
}

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

/* 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);
}

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

```
743 /* Devuelve la posición del puntero en registro X2. */
744 function quitar_puntero_aux2(var X1, var X2){
745     var V1 = U2_2(X1, X2);
746     return sacar_puntero(V1);
747 }
748
749 /* Devuelve el número entre el que hay
750  *  que dividir registro X2 para quitarle el puntero. */
751 function quitar_puntero_aux1(var X1, var X2){
752     var V1 = quitar_puntero_aux2(X1, X2);
753     var V2 = dos(X1, X2);
754     var V3 = uno(X1, X2);
755     return mult_veces(V1, V2, V3);
756 }
757
758 /* Devuelve el registro X2 sin puntero. */
759 function quitar_puntero(var X1, var X2){
760     var V1 = quitar_puntero_aux1(X1, X2);
761     var V2 = U2_2(X1, X2);
762     return div(V1, V2);
763 }
764
765 /* Devuelve el registro X2 con el
766  * puntero señalando a la posición X1. */
767 function saltar(var X1, var X2){
768     var V1 = U2_1(X1, X2);
769     var V2 = dos(X1, X2);
770     var V3 = quitar_puntero(X1, X2);
771     return mult_veces(V1, V2, V3);
772 }
773
774 /* ------------------------------------------------ */
775 /* Devuelve la instrucción
776  * en la primera posición de registro X2. */
777 function inst_uno(var X1, var X2){
778     var V1 = uno(X1, X2);
779     var V2 = U2_2(X1, X2);
780     return inst(V1, V2);
781 }
```

```
782
783 /* Devuelve la posición referida por la
784  * instrucción en la primera posición de registro X2. */
785 function v_pos_uno(var X1, var X2){
786     var V1 = inst_uno(X1, X2);
787     var V2 = U2_2(X1, X2);
788     return pos_inst(V1, V2);
789 }
790
791 /* Devuelve la posición referida por la
792  * instrucción X1 de registro X2. */
793 function v_pos(var X1, var X2){
794     var V1 = inst(X1, X2);
795     var V2 = U2_2(X1, X2);
796     return pos_inst(V1, V2);
797 }
798
799 /* Devuelve 1 si la posición X1 de registro X2
800  * es igual que la primera; 0 en caso contrario. */
801 function hay_salto(var X1, var X2){
802     var V1 = v_pos_uno(X1, X2);
803     var V2 = v_pos(X1, X2);
804     return es_igual(V1, V2);
805 }
806
807 /* Devuelve el registro X2 con el puntero
808  * señalando a la siguiente posición de la actual si
809  * la posición X1 es distinta de la primera posición;
810  * a la siguiente de la siguiente en caso contrario. */
811 function comparar(var X1, var X2){
812     var V1 = hay_salto(X1, X2);
813     var V2 = dos(X1, X2);
814     var V3 = avanzar_inst(X1, X2);
815     return mult_veces(V1, V2, V3);
816 }
817
818 /* ---------------------------------------------- */
819
```

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

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

/* 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);
}
```

```
859
860 /* Si X1 es 2 devuelve el registro X3
861  * después de ejecutar la instrucción 'borrar'
862  * en la posición X2; devuelve 0 si X1 no es 2. */
863 function eval_borrar(var X1, var X2, var X3){
864     var V1 = es_borrar(X1, X2, X3);
865     var V2 = eval_borrar_aux(X1, X2, X3);
866     return mult(V1, V2);
867 }
868
869 /* ---------------------------------------------- */
870
871 /* Devuelve el registro X3 después de ejecutar 'comparar'. */
872 function eval_comparar_aux(var X1, var X2, var X3){
873     var V1 = U3_2(X1, X2, X3);
874     var V2 = U3_3(X1, X2, X3);
875     return comparar(V1, V2);
876 }
877
878 /* Si el tipo de instrucción X1 es
879  * 'comparar', devuelve 1; 0 en caso contrario. */
880 function es_comparar(var X1, var X2, var X3){
881     var V1 = U3_1(X1, X2, X3);
882     var V2 = tres(X1, X2, X3);
883     return es_igual(V1, V2);
884 }
885
886 /* Si X1 es 3 devuelve el registro X3
887  * después de ejecutar la instrucción 'comparar'
888  * en la posición X2; devuelve 0 si X1 no es 3. */
889 function eval_comparar(var X1, var X2, var X3){
890     var V1 = es_comparar(X1, X2, X3);
891     var V2 = eval_comparar_aux(X1, X2, X3);
892     return mult(V1, V2);
893 }
894
895 /* ---------------------------------------------- */
896
```

```
/* 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);
}

/* Devuelve el registro después de ejecutar la intrucción
 * que indica X1 o el registro sin tocar si X1 es 0. */
function eval_registro(var X1, var X2, var X3){
    var V1 = eval_marcar(X1, X2, X3);
    var V2 = eval_borrar(X1, X2, X3);
    var V3 = eval_comparar(X1, X2, X3);
    var V4 = eval_saltar(X1, X2, X3);
    var V5 = eval_no_inst(X1, X2, X3);
    return suma(V1, V2, V3, V4, V5);
}

/* Comprueba si hay instrucción, ejecuta
 * en caso de haberla y vuelve a empezar el ciclo. */
function ejec3(var X1, var X2, var X3){
    var V1 = hay_inst(X1, X2, X3);
    var V2 = eval_registro(X1, X2, X3);
    return evaluar(V1, V2);
}

/* Saca el tipo de instrucción, la
 * posición referida y llama a «ejec3».
 * X1: instrucción.
 * X2: registro. */
function ejec2(var X1, var X2){

    var V1 = tipo_inst(X1, X2);
    var V2 = pos_inst(X1, X2);
    var V3 = U2_2(X1, X2);
    return ejec3(V1, V2, V3);
}

```

```
/* Saca la instrucción y llama a «ejec2» con ésta y el registro X2.
 * X1: puntero.
 * X2: registro. */
function ejec1(var X1, var X2){
    var V1 = inst(X1, X2);
    var V2 = U2_2(X1, X2);
    return ejec2(V1, V2);
}

/* 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);
    }
}

```

```
1013 /* Auxiliar para «computar». */
1014 function computar_aux(var X1, var X2){
1015     var V1 = uno(X1, X2);
1016     var V2 = mult(X1, X2);
1017     return evaluar(V1, V2);
1018 }
1019
1020 /*
1021  * Función recursiva primitiva que computa el mismo valor
1022  * que el programa C-- que recibe como argumento; en caso de termina
1023  * devuelve un registro con el puntero y programa después del cómput
1024  */
1025 function computar(var X1){
1026     var V1 = dos(X1);
1027     var V2 = U1_1(X1);
1028     return computar_aux(V1, V2);
1029 }
```