

```
1 #define function unsigned int
2 #define var unsigned int
3
4 /*****/
5     Funciones básicas.
6 /*****/
7 function cero(){
8     return 0;
9 }
10
11 function cero(var X1){
12     return 0;
13 }
14
15 function S(var X1){
16     X1++;
17     return X1;
18 }
19
20 function U1_1(var X1){
21     return X1;
22 }
23
24 function U2_1(var X1, var X2){
25     return X1;
26 }
27
28 function U2_2(var X1, var X2){
29     return X2;
30 }
31
32 function U3_1(var X1, var X2, var X3){
33     return X1;
34 }
35
36
37
```

```
38 function U3_2(var X1, var X2, var X3){
39     return X2;
40 }
41
42 function U3_3(var X1, var X2, var X3){
43     return X3;
44 }
45
46 function U4_1(var X1, var X2, var X3, var X4){
47     return X1;
48 }
49
50 function U4_2(var X1, var X2, var X3, var X4){
51     return X2;
52 }
53
54 function U4_3(var X1, var X2, var X3, var X4){
55     return X3;
56 }
57
58 function U4_4(var X1, var X2, var X3, var X4){
59     return X4;
60 }
61
62 function U5_1(var X1, var X2, var X3, var X4, var X5){
63     return X1;
64 }
65
66 function U5_2(var X1, var X2, var X3, var X4, var X5){
67     return X2;
68 }
69
70 function U5_3(var X1, var X2, var X3, var X4, var X5){
71     return X3;
72 }
73
74
75
```

```
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 }
```

```

114 /* Devuelve siempre 2. */
115 function dos(var X1, var X2, var X3){
116     var V1 = U3_2(X1, X2, X3);
117     return dos(V1);
118 }
119
120 /* Devuelve siempre 3. */
121 function tres(var X1, var X2){
122     var V1 = dos(X1, X2);
123     return S(V1);
124 }
125
126 /* Devuelve siempre 3. */
127 function tres(var X1, var X2, var X3){
128     var V1 = dos(X1, X2, X3);
129     return S(V1);
130 }
131
132 /* Devuelve siempre 4. */
133 function cuatro(var X1, var X2){
134     var V1 = tres(X1, X2);
135     return S(V1);
136 }
137
138 /* Devuelve siempre 4. */
139 function cuatro(var X1, var X2, var X3){
140     var V1 = tres(X1, X2, X3);
141     return S(V1);
142 }
143
144 /* ----- */
145
146 /* Función auxiliar para la función «suma». */
147 function suma_aux(var X1, var X2, var X3){
148     var V1 = U3_2(X1, X2, X3);
149     return S(V1);
150 }
151

```

```

152 /* Devuelve la suma de X1 y X2. */
153 function suma(var X1, var X2){
154     if(X1 == 0){
155         return U1_1(X2);
156     }
157     else{
158         X1--;
159         var rr = suma(X1, X2);
160         return suma_aux(X1, rr, X2);
161     }
162 }
163
164 /* ----- */
165
166 /* Auxiliar para «suma3». */
167 function suma_aux3(var X1, var X2, var X3){
168     var V1 = U3_2(X1, X2, X3);
169     var V2 = U3_3(X1, X2, X3);
170     return suma(V1, V2);
171 }
172
173 /* Devuelve la suma de X1, X2 y X3. */
174 function suma3(var X1, var X2, var X3){
175     var V1 = U3_1(X1, X2, X3);
176     var V2 = suma_aux3(X1, X2, X3);
177     return suma(V1, V2);
178 }
179
180 /* Auxiliar para «suma4». */
181 function suma_aux4(var X1, var X2, var X3, var X4){
182     var V1 = U4_2(X1, X2, X3, X4);
183
184     var V2 = U4_3(X1, X2, X3, X4);
185     var V3 = U4_4(X1, X2, X3, X4);
186     return suma3(V1, V2, V3);
187 }
188
189

```

```

190 /* Devuelve la suma de X1, X2, X3 y X4. */
191 function suma4(var X1, var X2, var X3, var X4){
192     var V1 = U4_1(X1, X2, X3, X4);
193     var V2 = suma_aux4(X1, X2, X3, X4);
194     return suma(V1, V2);
195 }
196
197 /* Auxiliar para «suma5». */
198 function suma_aux5(var X1, var X2, var X3, var X4, var X5){
199     var V1 = U5_2(X1, X2, X3, X4, X5);
200     var V2 = U5_3(X1, X2, X3, X4, X5);
201     var V3 = U5_4(X1, X2, X3, X4, X5);
202     var V4 = U5_5(X1, X2, X3, X4, X5);
203     return suma4(V1, V2, V3, V4);
204 }
205
206 /* Devuelve la suma de X1, X2, X3, X4 y X5. */
207 function suma5(var X1, var X2, var X3, var X4, var X5){
208     var V1 = U5_1(X1, X2, X3, X4, X5);
209     var V2 = suma_aux5(X1, X2, X3, X4, X5);
210     return suma(V1, V2);
211 }
212
213 /* ----- */
214
215 /* Función auxiliar para la función «resta».
216  * devuelve X1 - 1. */
217 function anterior(var X1){
218     if(X1 == 0){
219         return cero();
220     }
221
222     else{
223         X1--;
224         var rr = anterior(X1);
225         return U2_1(X1, rr);
226     }
227 }

```

```

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
260     return suma(V1, V2);
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

```



```

304 /* Devuelve 0 si X1 es mayor que 0 y 1 si es 0. */
305 function no_sg(var X1){
306     var V1 = uno(X1);
307     var V2 = U1_1(X1);
308     return resta(V1, V2);
309 }
310
311 /* Devuelve 0 si X1 es 0 y 1 si es mayor que 0. */
312 function sg(var X1){
313     var V1 = uno(X1);
314     var V2 = no_sg(X1);
315     return resta(V1, V2);
316 }
317
318 /* Devuelve 1 si X1 es menor o igual que X2;
319  * 0 en caso contrario. */
320 function menor_ig(var X1, var X2){
321     var V1 = resta(X1, X2);
322     return no_sg(V1);
323 }
324
325 /* Devuelve 1 si X1 es mayor o igual que X2;
326  * 0 en caso contrario. */
327 function mayor_ig(var X1, var X2){
328     var V1 = U2_2(X1, X2);
329     var V2 = U2_1(X1, X2);
330     return menor_ig(V1, V2);
331 }
332
333 /* Devuelve 1 si X1 es igual que X2; 0 en caso contrario. */
334 function igual(var X1, var X2){
335     var V1 = mayor_ig(X1, X2);
336     var V2 = menor_ig(X1, X2);
337     return mult(V1, V2);
338 }
339
340 /* ----- */
341

```

```

342 /* Declaración para funciones que usan recursión indirecta. */
343 function div_emp(var, var, var, var);
344
345 /* Función auxiliar para «div».
346  * Comprueba si dividendo es mayor que el divisor. */
347 function div_aux_mayor_ig(var X1, var X2, var X3){
348     var V1 = U3_2(X1, X2, X3);
349     var V2 = U3_3(X1, X2, X3);
350     return mayor_ig(V1, V2);
351 }
352
353 /* Función auxiliar para «div».
354  * Suma uno al resultado provisional. */
355 function div_aux_sig(var X1, var X2, var X3, var X4, var X5){
356     var V1 = U5_3(X1, X2, X3, X4, X5);
357     return S(V1);
358 }
359
360 /* Función auxiliar para «div».
361  * Resta divisor al dividendo. */
362 function div_aux_rt(var X1, var X2, var X3, var X4, var X5){
363     var V1 = U5_4(X1, X2, X3, X4, X5);
364     var V2 = U5_5(X1, X2, X3, X4, X5);
365     return resta(V1, V2);
366 }
367
368 /* Función auxiliar para «div».
369  * Suma uno al resultado, resta al
370  * dividendo el divisor y empieza el ciclo. */
371 function div_aux2(var X1, var X2, var X3, var X4, var X5){
372     var V1 = div_aux_sig(X1, X2, X3, X4, X5);
373
374     var V3 = div_aux_rt(X1, X2, X3, X4, X5);
375     var V2 = U5_5(X1, X2, X3, X4, X5);
376     return div_emp(V1, V2, V3);
377 }
378
379

```

```

380 /* Función auxiliar para «div».
381  * X1: ¿es dividendo mayor que divisor?
382  * X2: resultado provisional.
383  * X3: dividendo.
384  * X4: divisor. */
385 function div_aux1(var X1, var X2, var X3, var X4){
386     if(X1 == 0){
387         return U3_1(X2, X3, X4);
388     }
389     else{
390         X1--;
391         var rr = div_aux1(X1, X2, X3, X4);
392         return div_aux2(X1, rr, X2, X3, X4);
393     }
394 }
395
396 /* Función auxiliar para «div».
397  * X1: resultado provisional.
398  * X2: dividendo.
399  * X3: divisor. */
400 function div_emp(var X1, var X2, var X3){
401     var V1 = div_aux_mayor_ig(X1, X2, X3);
402     var V2 = U3_1(X1, X2, X3);
403     var V3 = U3_2(X1, X2, X3);
404     var V4 = U3_3(X1, X2, X3);
405     return div_aux1(V1, V2, V3, V4);
406 }
407
408 /* Devuelve n, donde  $n = X1/X2$ .
409  * X1: dividendo.
410  * X2: divisor. */
411 function div(var X1, var X2){
412     var V1 = cero(X1, X2);
413     var V2 = U2_1(X1, X2);
414     var V3 = U2_2(X1, X2);
415     return div_emp(V1, V2, V3);
416 }
417

```

```

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  * dividiendo 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

```

```

456 /* Devuelve 1 si X1 no es divisible
457    * entre X2; 0 en caso contrario. */
458 function no_es_div(var X1, var X2){
459     var V1 = es_div(X1, X2);
460     return no_sg(V1);
461 }
462
463 /* ----- */
464
465 /* Declaración para funciones que usan recursión indirecta. */
466 function vdiv_emp(var, var, var);
467
468 /* Función auxiliar para «vdiv».
469    * Devuelve 1 si es divisible X2 entre X3; si no 0. */
470 function vdiv_aux_es_div(var X1, var X2, var X3){
471     var V1 = U3_2(X1, X2, X3);
472     var V1 = U3_3(X1, X2, X3);
473     return es_div(V1, V2);
474 }
475
476 /* Función auxiliar para «vdiv».
477    * Divide X4 entre X5. */
478 function vdiv_aux_div(var X1,var X2,var X3,var X4,var X5){
479     var V1 = U5_4(X1, X2, X3, X4, X5);
480     var V2 = U5_5(X1, X2, X3, X4, X5);
481     return div(V1, V2);
482 }
483
484 /* Función auxiliar para «vdiv».
485    * Suma 1 al resultado provisional y divide
486    * dividendo entre divisor. */
487 function vdiv_aux2(var X1, var X2, var X3, var X4,var X5){
488     var V1 = div_aux_sig(X1, X2, X3, X4, X5);
489     var V2 = vdiv_aux_div(X1, X2, X3, X4, X5);
490     var V3 = U5_5(X1, X2, X3, X4, X5);
491     return vdiv_emp(V1, V2, V2);
492 }
493

```

```

494 /* Función auxiliar para «vdiv». Devuelve el resultado
495  * si X3 no es divisible entre X4; si no, sigue el ciclo. */
496 function vdiv_aux1(var X1, var X2, var X3, var X4){
497     if(X1 == 0){
498         return U3_1(X2, X3, X4);
499     }
500     else{
501         X1--;
502         var rr = vdiv_aux1(X1, X2, X3, X4);
503         return vdiv_aux2(X1, rr, X2, X3, X4);
504     }
505 }
506
507 /* Función auxiliar para «vdiv». Comienza el ciclo. */
508 function vdiv_emp(var X1, var X2, var X3){
509     var V1 = vdiv_aux_es_div(X1, X2, X3);
510     var V2 = U3_1(X1, X2, X3);
511     var V3 = U3_2(X1, X2, X3);
512     var V4 = U3_3(X1, X2, X3);
513     return vdiv_aux1(V1, V2, V3, V4);
514 }
515
516 /* Devuelve las veces que es divisible X1 entre X2. */
517 function vdiv(var X1, var X2){
518     var V1 = cero(X1, X2);
519     var V2 = U2_1(X1, X2);
520     var V3 = U2_2(X1, X2);
521     return vdiv_emp(V1, V2, V3);
522 }
523
524 /* ----- */
525
526 /* Función auxiliar para «nd_hasta_aux2».
527  * Suma 2 para que no se compruebe si es divisible entre 0 o 1.*/
528 function nd_hasta_aux3(var X1, var X2, var X3){
529     var V1 = U3_1(X1, X2, X3);
530     var V2 = dos(X1, X2, X3);
531     return suma(V1, V2);
532 }

```

```

532 /* Función auxiliar para «nd_hasta_aux1». */
533 function nd_hasta_aux2(var X1, var X2, var X3){
534     var V1 = U3_3(X1, X2, X3);
535     var V2 = nd_hasta_aux3(X1, X2, X3);
536     return no_es_div(V1, V2);
537 }
538
539 /* Función auxiliar para «no_div_hasta». */
540 function nd_hasta_aux1(var X1, var X2, var X3){
541     var V2 = nd_hasta_aux2(X1, X2, X3);
542     var V1 = U3_2(X1, X2, X3);
543     return mult(V1, V2);
544 }
545
546 /* Función auxiliar para «es_primo».
547  * Devuelve 1 si no hay divisor de X2 entre 2 y X1 + 1. */
548 function no_div_hasta(var X1, var X2){
549     if(X1 == 0){
550         return uno(X2);
551     }
552     else{
553         X1--;
554         var rr = no_div_hasta(X1, X2);
555         return nd_hasta_aux1(X1, rr, X2);
556     }
557 }
558
559 /* Función auxiliar para «es_primo». */
560 function partir(var X1){
561     var V1 = anterior(X1);
562     var V2 = dos(X1);
563
564     return div(V1, V2);
565 }
566
567
568
569

```

```

570 /* Devuelve 1 si X1 es primo; 0 en caso contrario. */
571 function es_primo(var X1){
572     var V1 = partir(X1);
573     var V2 = U1_1(X1);
574     return no_div_hasta(V1, V2);
575 }
576
577 /* Devuelve 0 si X1 es primo; 1 en caso contrario. */
578 function no_es_primo(var X1){
579     var V1 = es_primo(X1);
580     return no_sg(V1);
581 }
582
583 /* ----- */
584
585 /* Declaración de función. */
586 function sig_primo_desde(var);
587
588 /* Función auxiliar para «sig_primo_aux1». */
589 function sig_primo_aux2(var X1, var X2, var X3){
590     var V1 = U3_2(X1, X2, X3);
591     return sig_primo_desde(V1);
592 }
593
594 /* Si X2 es primo lo devuelve; si no
595  * sigue buscando el menor primo mayor que X2. */
596 function sig_primo_aux1(var X1, var X2){
597     if(X1 == 0){
598         return U1_1(X2);
599     }
600     else{
601
602         X1--;
603         var rr = sig_primo_aux1(X1, X2);
604         return sig_primo_aux2(X1, rr, X2);
605     }
606 }
607

```



```

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 /* ----- */
641 /* Devuelve el primo correspondiente a la posición X1.
642  * X1: posición. */
643 function primo_num_inst(var X1, var X2){
644     var V1 = U2_1(X1, X2);
645     return inst_primo(V1);
646 }

```

```

646 /* Devuelve la instrucción en la posición X1.
647  * X1: posición.
648  * X2: registro. */
649 function inst(var X1, var X2){
650     var V1 = U2_2(X1, X2);
651     var V2 = primo_num_inst(X1, X2);
652     return vdiv(V1, V2);
653 }
654
655 /* Devuelve 1 si X1 es instrucción; 0 en caso contrario.
656  * X1: instrucción. */
657 function hay_inst(var X1, var X2){
658     var V1 = U2_1(X1, X2);
659     return sg(V1);
660 }
661
662 /* Devuelve: 0 si no hay instrucción o ésta es '1';
663  * 1 si '0'.
664  * 2 si '='.
665  * 3 si '*'. */
666 function tipo_inst_aux(var X1, var X2){
667     var V1 = U2_1(X1, X2);
668     var V2 = cuatro(X1, X2);
669     return resto(V1, V2);
670 }
671
672 /* Devuelve, según la instrucción en X1:
673  * 0 si no hay instrucción.
674  * 1 si '1'.
675  * 2 si '0'.
676  * 3 si '='.
677  * 4 si '*'. */
678 function tipo_inst(var X1, var X2){
679     var V1 = tipo_inst_aux(X1, X2);
680     var V2 = hay_inst(X1, X2);
681     return suma(V1, V2);
682 }
683

```

```

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

```

```

722 /* Devuelve el registro X2 modificado con una marca más
723  * en la posición X1, o un símbolo «1» si estaba vacía,
724  * y el puntero señalando a la siguiente posición. */
725 function marcar(var X1, var X2){
726     var V1 = cuatro(X1, X2);
727     var V2 = primo_num_inst(X1, X2);
728     var V3 = avanzar_inst(X1, X2);
729     return mult_veces(V1, V2, V3);
730 }
731 /* ----- */
732
733 /* Auxiliar para «borrar_aux2». Resta para mantener
734  * el símbolo en la posición referida si éste no es '1'. */
735 function borrar_aux3(var X1, var X2){
736     var V1 = U2_1(X1, X2);
737     var V2 = tipo_inst_aux(X1, X2);
738     return resta(V1, V2);
739 }
740
741 /* Auxiliar para «borrar_aux».
742  * X1: instrucción en posición referida.
743  * X2: primo en posición referida. */
744 function borrar_aux2(var X1, var X2){
745     var V1 = borrar_aux3(X1, X2);
746     var V2 = U2_2(X1, X2);
747     var V2 = uno(X1, X2);
748     return mult_veces(V1, V2, V3);
749 }
750
751 /* Devuelve el número entre el que hay que dividir
752  * el registro X2 para borrar la posición X1.
753  * X1: posición referida.
754  * X2: registro. */
755 function borrar_aux1(var X1, var X2){
756     var V1 = inst(X1, X2);
757     var V2 = primo_num_inst(X1, X2);
758     return borrar_aux2(V1, V2);
759 }

```

```

760 /* Devuelve el registro con la posición X1 borrada.
761  * X1: posición referida.
762  * X2: registro. */
763 function borrar(var X1, var X2){
764     var V1 = avanzar_inst(X1, X2);
765     var V2 = borrar_aux1(X1, X2);
766     return div(V1, V2);
767 }
768
769 /* ----- */
770
771
772 /* Devuelve la posición del puntero en registro X2. */
773 function quitar_puntero_aux2(var X1, var X2){
774     var V1 = U2_2(X1, X2);
775     return sacar_puntero(V1);
776 }
777
778 /* Devuelve el número entre el que hay
779  * que dividir registro X2 para quitarle el puntero. */
780 function quitar_puntero_aux1(var X1, var X2){
781     var V1 = quitar_puntero_aux2(X1, X2);
782     var V2 = dos(X1, X2);
783     var V3 = uno(X1, X2);
784     return mult_veces(V1, V2, V3);
785 }
786
787 /* Devuelve el registro X2 sin puntero. */
788 function quitar_puntero(var X1, var X2){
789     var V1 = quitar_puntero_aux1(X1, X2);
790     var V2 = U2_2(X1, X2);
791
792     return div(V1, V2);
793 }
794
795
796
797

```

```

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

```

```

874 /* Si X1 es 1 devuelve el registro X3
875  * después de ejecutar la instrucción 'marcar'
876  * en la posición X2; devuelve 0 si X1 no es 1. */
877 function eval_marcar(var X1, var X2, var X3){
878     var V1 = es_marcar(X1, X2, X3);
879     var V2 = eval_marcar_aux(X1, X2, X3);
880     return mult(V1, V2);
881 }
882
883 /* ----- */
884
885 /* Devuelve el registro X3 después de ejecutar 'borrar'. */
886 function eval_borrar_aux(var X1, var X2, var X3){
887     var V1 = U3_2(X1, X2, X3);
888     var V2 = U3_3(X1, X2, X3);
889     return borrar(V1, V2);
890 }
891
892 /* Si el tipo de instrucción X1 es
893  * 'borrar', devuelve 1; 0 en caso contrario. */
894 function es_borrar(var X1, var X2, var X3){
895     var V1 = U3_1(X1, X2, X3);
896     var V2 = dos(X1, X2, X3);
897     return es_igual(V1, V2);
898 }
899
900 /* Si X1 es 2 devuelve el registro X3
901  * después de ejecutar la instrucción 'borrar'
902  * en la posición X2; devuelve 0 si X1 no es 2. */
903 function eval_borrar(var X1, var X2, var X3){
904     var V1 = es_borrar(X1, X2, X3);
905
906     var V2 = eval_borrar_aux(X1, X2, X3);
907     return mult(V1, V2);
908 }
909
910 /* ----- */
911

```



```

912 /* Devuelve el registro X3 después de ejecutar 'comparar'. */
913 function eval_comparar_aux(var X1, var X2, var X3){
914     var V1 = U3_2(X1, X2, X3);
915     var V2 = U3_3(X1, X2, X3);
916     return comparar(V1, V2);
917 }
918
919 /* Si el tipo de instrucción X1 es
920  * 'comparar', devuelve 1; 0 en caso contrario. */
921 function es_comparar(var X1, var X2, var X3){
922     var V1 = U3_1(X1, X2, X3);
923     var V2 = tres(X1, X2, X3);
924     return es_igual(V1, V2);
925 }
926
927 /* Si X1 es 3 devuelve el registro X3
928  * después de ejecutar la instrucción 'comparar'
929  * en la posición X2; devuelve 0 si X1 no es 3. */
930 function eval_comparar(var X1, var X2, var X3){
931     var V1 = es_comparar(X1, X2, X3);
932     var V2 = eval_comparar_aux(X1, X2, X3);
933     return mult(V1, V2);
934 }
935 /* ----- */
936 /* Devuelve el registro X3 después de ejecutar 'saltar'. */
937 function eval_saltar_aux(var X1, var X2, var X3){
938     var V1 = U3_2(X1, X2, X3);
939     var V2 = U3_3(X1, X2, X3);
940     return saltar(V1, V2);
941 }
942
943 /* Si el tipo de instrucción X1 es 'saltar', devuelve 1;
944  * 0 en caso contrario. */
945 function es_saltar(var X1, var X2, var X3){
946     var V1 = U3_1(X1, X2, X3);
947     var V2 = cuatro(X1, X2, X3);
948     return es_igual(V1, V2);
949 }

```

```

950 /* Si X1 es 4 devuelve el registro X3
951  * después de ejecutar la instrucción 'saltar'
952  * en la posición X2; devuelve 0 si X1 no es 4. */
953 function eval_saltar(var X1, var X2, var X3){
954     var V1 = es_saltar(X1, X2, X3);
955     var V2 = eval_saltar_aux(X1, X2, X3);
956     return mult(V1, V2);
957 }
958
959
960 /* ----- */
961
962
963 /* Función auxiliar para «eval_no_inst». */
964 function eval_no_inst_aux(var X1, var X2, var X3){
965     var V1 = U3_1(X1, X2, X3);
966     return no_sg(V1);
967 }
968
969 /* Devuelve el registro X3 sin modificar
970  * si el tipo de instrucción X1 es 0;
971  * devuelve 0 en caso contrario. */
972 function eval_no_inst(var X1, var X2, var X3){
973     var V1 = no_inst_aux(X1, X2, X3);
974     var V2 = U3_3(X1, X2, X3);
975     return mult(V1, V2);
976 }
977
978
979 /* ----- */
980
981
982 /* Devuelve 1 si hay instrucción en X1; 0 en caso contrario . */
983 function hay_inst(var X1, var X2, var X3){
984     var V1 = U3_1(X1, X2, X3);
985     return sg(V1);
986 }
987

```

```

988 /* Devuelve el registro después de ejecutar la instrucción
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 }

```

```

1026 /* Toma el registro X1 y llama a «ejec1»
1027  * con éste y el puntero contenido en él. */
1028 function ejecutar_inst(var X1){
1029     var V1 = sacar_puntero(X1);
1030     var V2 = U1_1(X1);
1031     return ejec1(V1, V2);
1032 }
1033
1034 /* X3: registro. */
1035 function ejecutar(var X1, var X2, var X3){
1036     var V1 = U3_3(X1, X2, X3);
1037     return ejecutar_inst(V1);
1038 }
1039
1040 /*
1041  * Si X1 es 0 devuelve el registro y acaba la computación;
1042  * si es 1 ejecuta la siguiente instrucción, si la hubiera,
1043  * en registro X2; es decir, ejecuta el siguiente paso.
1044  */
1045 function evaluar(var X1, var X2){
1046     if(X1 == 0){
1047         return U1_1(X2);
1048     }
1049     else{
1050         X1--;
1051         var rr = evaluar(X1, X2);
1052         return ejecutar(X1, rr, X2);
1053     }
1054 }
1055
1056 /* Auxiliar para «computar». */
1057 function computar_aux(var X1, var X2){
1058     var V1 = uno(X1, X2);
1059     var V2 = mult(X1, X2);
1060     return evaluar(V1, V2);
1061 }
1062
1063

```

```
1064 /*
1065  * Función recursiva primitiva que computa el mismo valor
1066  * que el programa C-- que recibe como argumento; en caso de termina
1067  * devuelve un registro con el puntero y programa después del cómputo
1068  */
1069 function computar(var X1){
1070     var V1 = dos(X1);
1071     var V2 = U1_1(X1);
1072     return computar_aux(V1, V2);
1073 }
```