```c
#include <stdio.h>
#include <stdlib.h>

void menu();
int* Zapoln (int*, int, int, int);
void print (int*, int);
int* Plus (int*, int, int, int, struct condition);
struct condition moving (struct condition, int*, int);
void printS (struct condition, int*, int);
int* multiplication (int*, int, int, struct condition);
int* Zapol (int*, int, int);
void printZ (int*, int);
struct condition startN (struct condition, int*, int);
struct condition startM (struct condition, int*, int);
int* Subtraction (int*, int, int, struct condition);
int* Division (int*, int, int, struct condition);
void printD (int*, int, struct condition);


struct condition
{
    int* p;
    int zn;
    char s;
    int sost;
};

void printS (struct condition q, int* L, int size)
{
    int *l;

    printf ("q%d %d %c\n", q.sost, q.zn, q.s);

    for (l = L; l < size + L; l++)
    {
        if (q.p == l) printf ("q");
        else printf (" ");
    }

    printf("\n");

    for (l = L; l < size + L; l++)
    {
        printf ("%d", *l);
    }

    printf("\n");
    printf("\n");
}

int* Zapoln (int* L, int n, int m, int size)
{
    int* l;

    for (l = L; l < L + size; l++)
        *l = 0;

    l = L + n + 5;

    while (n)
    {
        *l = 1;
        l++;
        n--;
    }

```

```
 67      l++;
 68
 69      while (m)
 70      {
 71          *l = 1;
 72          l++;
 73          m--;
 74      }
 75
 76      return L;
 77  }
 78
 79  void menu()
 80  {
 81      puts ("menu:");
 82      puts ("1: first + second");
 83      puts ("2: first * 2");
 84      puts ("3: second * 2");
 85      puts ("4: first - 2");
 86      puts ("5: second - 2");
 87      puts ("6: first / 2");
 88      puts ("7: second / 2");
 89      puts ("other - end of work");
 90  }
 91
 92  void print (int* L, int size)
 93  {
 94      int *l;
 95      int c = 0;
 96
 97      for (l = L; l < size + L; l++)
 98          if (*l == 1)
 99              c++;
100      printf ("\n%d", c);
101
102  }
103
104  struct condition moving (struct condition q, int*L, int
size)
105  {
106      if (q.s == 'L')
107      {
108          while (q.zn == *(q.p) && (q.p) != L && q.p < L +
size - 1)
109          {
110              (q.p)--;
111              printS (q, L, size);
112          }
113      }
114
115      if (q.s == 'R')
116      {
117          while (q.zn == *(q.p) && (q.p) != L && q.p < L +
size - 1)
118          {
119              (q.p)++;
120              printS (q, L, size);
121          }
122      }
123
124      return q;
125  }
126
127  struct condition startN (struct condition q, int*L, int
nach)
128  {
```

```
129       q.sost = 1;
130       q.s = 'N';
131
132       q.p = L + nach + 5;
133       q.zn = 1;
134
135       while (*(q.p))
136           (q.p)++;
137
138       (q.p)++;
139
140       while (*(q.p))
141           (q.p)++;
142
143       (q.p)--;
144       return q;
145   }
146
147   int* Plus (int* L, int nach, int m, int size, struct
condition q)
148   {
149
150       if (q.sost == 0) q = startN (q, L, nach);
151       printS (q, L, size);
152
153       q.s = 'L';
154       printS (q, L, size);
155       q = moving (q, L, size);
156
157       q.s = 'N';
158       q.zn = 0;
159       printS (q, L, size);
160
161       *(q.p) = 1;
162       q.sost ++;
163       q.zn = 1;
164       printS (q, L, size);
165
166       q.s = 'L';
167       q = moving (q, L, size);
168
169
170       q.zn = 0;
171       q.s = 'N';
172       printS (q, L, size);
173
174       q.s = 'R';
175       q.sost ++;
176       printS (q, L, size);
177
178       q = moving (q, L, size);
179       q.s = 'N';
180       q.zn = 1;
181       printS (q, L, size);
182
183       q.s = 'N';
184       q.sost ++;
185       *(q.p) = 0;
186       q.zn = 0;
187       printS (q, L, size);
188
189       q.s = 'R';
190       printS (q, L, size);
191
192       q = moving (q, L, size);
193
```

```
194       q.zn = 1;
195       printS (q, L, size);
196
197       q = moving (q, L, size);
198       printS (q, L, size);
199
200       q.s = 'N';
201       q.zn = 0;
202       printS (q, L, size);
203
204       q.zn = 0;
205       q.sost = 0;
206       q.s = 'L';
207       printS (q, L, size);
208
209       q = moving (q, L, size);
210
211       q.zn = 1;
212       q.s = 'N';
213       printS (q, L, size);
214
215       return L;
216   }
217
218   int* Zapol (int* L, int first, int size)
219   {
220       int* l;
221       int n, m;
222
223       for (l = L; l < L + size; l++)
224           *l = 0;
225
226       n = abs(first);
227       l = L + n + 5;
228       n*= 2;
229       if (first < 0) n--;
230
231       while (n)
232       {
233           *l = 1;
234           l++;
235           n--;
236       }
237
238       return L;
239   }
240
241   void printZ (int* L, int size)
242   {
243       int *l;
244       int c = 0;
245
246       for (l = L; l < size + L; l++)
247           if (*l == 1)
248               c++;
249       if (c % 2 == 1) c = (c+1) * (-1);
250       printf ("\n%d", c/2);
251   }
252
253   struct condition startM (struct condition q, int*L, int
nach)
254   {
255       q.sost = 1;
256       q.s = 'N';
257
258       q.p = L;
```

```
259       q.zn = 1;
260
261       while (!*(q.p))
262           (q.p)++;
263
264       while (*(q.p))
265           (q.p)++;
266
267       (q.p)--;
268
269       return q;
270   }
271
272   int* multiplication (int* L, int n, int size, struct
condition q)
273   {
274       int flag = 1;
275       int i;
276
277       L = Zapol (L, n, size);
278
279       if (n == 0)
280           return L;
281
282       q = startM (q, L, abs(n) + 5);
283       printS (q, L, size);
284
285       q.sost++;
286       q.s = 'L';
287       q.zn = 1;
288       i = q.sost;
289
290       while (*(q.p) == 1)
291       {
292           q.sost = i;
293           (q.p)--;
294           printS (q, L, size);
295           if (*(q.p) == 0) break;
296
297           q.sost ++;
298           (q.p)--;
299           printS (q, L, size);
300       }
301
302       if (q.sost == 3)
303       {
304           q.sost = 4;
305           q.s = 'R';
306           q.zn = 0;
307           q = moving (q, L, size);
308
309           q.zn = 1;
310           q = moving (q, L, size);
311
312           q.sost++;
313           q.zn = 0;
314           q.s = 'L';
315           (q.p)--;
316           printS (q, L, size);
317       }
318       else
319       {
320           q.sost = 6;
321           q.s = 'R';
322           q.zn = 0;
323           (q.p)++;
```

```
324             printS (q, L, size);
325
326             q.zn = 1;
327             q = moving (q, L, size);
328
329             q.sost++;
330             q.zn = 1;
331             q.s = 'N';
332             *(q.p) = 1;
333             printS (q, L, size);
334
335             q.s = 'L';
336             q.p--;
337             q.sost = 5;
338             printS (q, L, size);
339         }
340
341         while (1)
342         {
343         q.sost = 8;
344         q.zn = 0;
345         q.s = 'N';
346         *(q.p) = 0;
347
348         q.sost ++;
349         q.s = 'R';
350         q.p++;
351         printS (q, L, size);
352
353         q.zn = 1;
354         q = moving (q, L, size);
355
356         q.sost ++;
357         q.zn = 1;
358         q.s = 'N';
359         *(q.p) = 1;
360         printS (q, L, size);
361
362         q.s = 'R';
363         q = moving (q, L, size);
364
365         q.sost ++;
366         q.zn = 1;
367         q.s = 'N';
368         *(q.p) = 1;
369         printS (q, L, size);
370
371         q.s = 'L';
372         q = moving (q, L, size);
373
374         q.sost ++;
375         q.zn = 0;
376         q.s = 'L';
377         q = moving (q, L, size);
378         if (q.p == L) break;
379         }
380
381         return L;
382     }
383
384     int* Subtraction (int* L, int n, int size, struct condition
    g)
385     {
386
387         L = Zapol (L, n, size);
388
```

```
389       q = startM (q, L, abs(n) + 5);
390       printS (q, L, size);
391
392       q.sost ++;
393       q.zn = 1;
394       q.s = 'L';
395       q.p --;
396       printS (q, L, size);
397
398       q.sost ++;
399       q.zn = 1;
400       q.s = 'L';
401       q.p --;
402       printS (q, L, size);
403
404       if (*(q.p) == 0)
405       {
406           q.sost ++;
407           q.zn = 0;
408           q.s = 'R';
409           q.p++;
410           printS (q, L, size);
411
412           q.sost ++;
413           q.zn = 0;
414           q.s = 'N';
415           *(q.p) = 0;
416           printS (q, L, size);
417
418           q.sost = 0;
419           q.zn = 0;
420           q.s = 'R';
421           q.p++;
422           printS (q, L, size);
423       }
424       else
425       {
426           q.sost = 6;
427           q.zn = 0;
428           q.s = 'R';
429           q.p++;
430           printS (q, L, size);
431
432           q.zn = 1;
433           q = moving (q, L, size);
434
435           q.s = 'L';
436           q.zn = 0;
437           while (q.sost != 10)
438           {
439               q.sost++;
440               *(q.p) = 0;
441               q.p--;
442               printS (q, L, size);
443           }
444
445           q.sost = 0;
446           *(q.p) = 0;
447           q.p--;
448           printS (q, L, size);
449       }
450
451       return L;
452   }
453
454   void printD (int* L, int size, struct condition q)
```

```
{
    int *l;
    int c = 0, d = 0, i;

    l = L;
    while (*l == 0 && l < L + size - 1) l++;

    while (*l == 1 && l < L + size - 1)
    {
            l++;
            d++;
    }

    while (*l == 0 && l < L + size - 1) l++;

    while (*l == 1 && l < L + size - 1)
    {
            l++;
            c++;
    }

    if ((q.p) == L || (*(q.p) == 1 && *(q.p-1) == 0))
    {
        c = d;
        d = 0;
    }


    if (c % 2 == 1) c = (c+1) * (-1);
    if (d % 2 == 1) d = (d+1) * (-1);

    printf ("\n%d %d", c/2, d/2);
}

int* Division (int* L, int n, int size, struct condition q)
{

    L = Zapol (L, n, size);
    int flag = 0;

    q = startM (q, L, abs(n) + 5);
    printS (q, L, size);
    q.s = 'L';

    while (1)
    {
        q.sost = 2;
        q.p --;
        printS (q, L, size);
        if (*(q.p) == 0) break;

        q.sost ++;
        q.p --;
        printS (q, L, size);
        if (*(q.p) == 0) break;

        q.sost ++;
        q.p --;
        printS (q, L, size);
        if (*(q.p) == 0) break;

        q.sost ++;
        q.p --;
        printS (q, L, size);
        if (*(q.p) == 0) break;
    }
```

```
521
522         switch (q.sost)
523             {
524                 case 3:
525                 flag = 1;
526
527                 q.sost = 22;
528                 q.zn = 0;
529                 q.s = 'L';
530                 q.p --;
531                 printS (q, L, size);
532
533                 q.sost = 23;
534                 q.zn = 1;
535                 q.s = 'L';
536                 *(q.p) = 1;
537                 q.p --;
538                 printS (q, L, size);
539
540                 q.sost = 24;
541                 q.zn = 1;
542                 q.s = 'R';
543                 *(q.p) = 1;
544                 q.p ++;
545                 printS (q, L, size);
546
547                 q = moving (q, L, size);
548
549                 case 2:
550                 if (flag == 0)
551                 {
552                 q.sost = 18;
553                 q.zn = 0;
554                 q.s = 'L';
555                 q.p --;
556                 printS (q, L, size);
557
558                 q.sost = 19;
559                 q.zn = 1;
560                 q.s = 'R';
561                 *(q.p) = 1;
562                 q.p ++;
563                 printS (q, L, size);
564
565                 q.sost = 20;
566                 q.zn = 0;
567                 q.s = 'R';
568                 *(q.p) = 0;
569                 q.p ++;
570                 printS (q, L, size);
571                 }
572
573                 q.sost = 21;
574                 q.zn = 0;
575                 q.s = 'R';
576                 *(q.p) = 0;
577                 q.p ++;
578                 printS (q, L, size);
579
580                 *(q.p) = 0;
581
582                 case 4:;
583
584                 case 5:
585
586                 q.sost = 6;
```

```
587                 q.zn = 0;
588                 q.s = 'R';
589                 q.p ++;
590                 printS (q, L, size);
591
592                 q.zn = 1;
593                 q = moving (q, L, size);
594
595                 q.sost++;
596                 q.zn = 0;
597                 q.s = 'L';
598                 q.p --;
599                 if (*(q.p) == 0)
600                 {
601                     printD (L, size, q);
602                     return L;
603                 }
604                 printS (q, L, size);
605
606                 q.sost++;
607                 q.zn = 1;
608                 q.s = 'L';
609                 q.p --;
610                 printS (q, L, size);
611
612                 q.sost ++;
613                 q.zn = 0;
614                 q.s = 'L';
615                 *(q.p) = 0;
616                 q.p --;
617                 printS (q, L, size);
618
619                 while (1)
620                 {
621
622                     q.sost = 10;
623                     q.zn = 1;
624                     q.s = 'L';
625                     *(q.p) = 0;
626                     q.p --;
627                     printS (q, L, size);
628
629                     if (*q.p == 1)
630                     {
631                     q.sost++;
632                     q.zn = 0;
633                     q.s = 'R';
634                     *(q.p) = 0;
635                     q.p ++;
636                     printS (q, L, size);
637
638                     q = moving (q, L, size);
639
640                     q.sost++;
641                     q.zn = 0;
642                     q.s = 'R';
643                     while (*q.p == 1) q.p++;
644
645                     q.sost++;
646                     q.zn = 1;
647                     q.s = 'L';
648                     *(q.p) = 1;
649                     q.p --;
650                     printS (q, L, size);
651
652                     q = moving (q, L, size);
```

```
653
654                     q.sost++;
655                     q.zn = 0;
656                     q.s = 'L';
657                     *(q.p) = 0;
658                     q.p --;
659                     printS (q, L, size);
660
661                     q = moving (q, L, size);
662
663                     if (q.p == L)
664                     {
665                         printD (L, size, q);
666                         break;
667                     }
668                     }
669                     else
670                     {
671                         q.sost = 15;
672                         q.zn = 0;
673                         q.s = 'L';
674                         *(q.p) = 0;
675                         q.p --;
676                         printS (q, L, size);
677
678                         q.sost = 16;
679                         q.zn = 0;
680                         q.s = 'R';
681                         q = moving (q, L, size);
682
683                         q.sost = 17;
684                         q.zn = 1;
685                         q.s = 'R';
686                         q = moving (q, L, size);
687
688                         q.sost = 0;
689                         q.p--;
690                         printS (q, L, size);
691                         printD (L, size, q);
692                         break;
693                     }
694                 }
695
696                 default: return L;
697             };
698
699     return L;
700 }
701
702 int main()
703 {
704     int n,m, size, small, big;
705     struct condition q;
706
707     printf ("Vvedite chisla:\n");
708     scanf ("%d%d", &n, &m);
709
710     if (abs(n) < abs(m))
711     {
712         small = abs(n) + 1;
713         big = abs(m);
714     }
715     else
716     {
717         big = abs(n);
718         small = abs(m) + 1;
```

```
719         }
720
721         int *lenta;
722
723         size = (2*big + small)*4;
724
725         lenta = (int*)malloc(sizeof(int)*size);
726         if (lenta == NULL)
727             return -1;
728
729         int option;
730
731         do
732         {
733             q.sost = 0;
734             menu();
735             printf ("\n");
736             scanf ("%d", &option);
737             printf ("\n");
738
739             switch (option)
740             {
741                 case 1:
742                 if (n > 0 && m > 0)
743                 {
744                     lenta = Zapoln (lenta, n, m, size);
745                     lenta = Plus (lenta, n, m, size, q);
746                     print (lenta, size);
747                 }
748                 else printf ("Only natural numbers");
749                 break;
750
751                 case 2:
752                 lenta = multiplication (lenta, n, size, q);
753                 printZ (lenta, size);
754                 break;
755
756                 case 3:
757                 lenta = multiplication (lenta, m, size, q);
758                 printZ (lenta, size);
759                 break;
760
761                 case 4:
762                 if (n > 0)
763                 {
764                     lenta = Subtraction (lenta, n, size, q);
765                     printZ (lenta, size);
766                 }
767                 else printf ("Only natural numbers");
768                 break;
769
770                 case 5:
771                 if (m > 0)
772                 {
773                     lenta = Subtraction (lenta, m, size, q);
774                     printZ (lenta, size);
775                 }
776                 else printf ("Only natural numbers");
777                 break;
778
779                 case 6:
780                 lenta = Division (lenta, n, size, q);
781                 break;
782
783                 case 7:
784                 lenta = Division (lenta, m, size, q);
```

```
785                 break;
786
787                 default: option = 0;
788             };
789
790             printf ("\n");
791             printf ("\n");
792             printf ("Ishod: %d %d\n", n, m);
793
794         } while (option != 0);
795     }
```