Programmation en C

Corrigé TD8

```
 1 /* Rationnels
 2  ---------------------------------------------- */
 3 #include <stdlib.h>
 4 #include <stdio.h>
 5
 6 typedef struct {
 7   int num;
 8   int den;
 9 } rationnel;
10
11 rationnel UN = {1,1};
12
13 rationnel init(int a,int b) {
14   rationnel r;
15   r.num = a;
16   r.den = b;
17   return r;
18 }
19
20 rationnel addition(rationnel r1, rationnel r2) {
21   rationnel s;
22
23   s.num = r1.num * r2.den + r2.num * r1.den;
24   s.den = r1.den * r2.den;
25   return s;
26 }
27
28 rationnel multiplication(rationnel r1, rationnel r2) {
29   rationnel m;
```

```
30
31   m.num = r1.num * r2.num;
32   m.den = r1.den * r2.den;
33   return m;
34 }
35
36 void affiche(rationnel r) {
37   printf("%d / %d ",r.num, r.den);
38 }
39
40 int pgcd(int a, int b) {
41 if (b==0) return a;
42 return pgcd(b,a%b);
43 }
44
45 void simplification(rationnel *r) {
46   int p = pgcd(r->num,r->den);
47   r->num = r->num / p;
48   r->den = r->den / p;
49 }
50
51 rationnel *addition1(rationnel r1, rationnel r2) {
52   rationnel *s = malloc(sizeof(rationnel));
53   rationnel r = addition(r1,r2);
54   s->num = r.num;
55   s->den = r.den;
56   simplification(s);
57   return s;
58 }
59
60 rationnel *multiplication1(rationnel r1, rationnel r2) {
61   rationnel *m = malloc(sizeof(rationnel));
62   rationnel k = multiplication(r1,r2);
63   m->num = k.num;
64   m->den = k.den;
65   simplification(m);
66   return m;
67 }
```

```
68
69 int main (int argc, char *argv[]) {
70   rationnel r1,r2;
71   rationnel r3;
72   rationnel r4;
73   rationnel *r5;
74   rationnel *r6;
75   r1 = init(atoi(argv[1]),atoi(argv[2]));
76   r2 = init(atoi(argv[3]),atoi(argv[4]));
77   affiche(r1);printf("\n");
78   affiche(r2);printf("\n");
79
80   r3 = addition(r1,r2);
81   r4 = multiplication(r1,r2);
82   affiche(r3);printf("\n");
83   simplification(&r3);
84   affiche(r3);printf("\n");
85   affiche(r4);printf("\n");
86
87   r5=addition1(r1,r2);
88   r6=multiplication1(r1,r2);
89   affiche(*r5);printf("\n");
90   simplification(r5);
91   affiche(*r5);printf("\n");
92   affiche(*r6);printf("\n");
93
94
95   return EXIT_SUCCESS;
96 }
```

```
 1 /* Grand Nombre et Factoriel
 2  ------------------------------------------------ */
 3 #include <stdio.h>
 4 #include <stdlib.h>
 5 #define N 100
 6
 7 typedef struct {
 8   int *val;
```

```
 9   int taille;
10 } entierlong;
11
12 /* on suppose a<1000 */
13 entierlong *set(int a){
14   int i;
15
16   entierlong *e = malloc(sizeof(entierlong));
17   (*e).val=malloc(100*sizeof(int));
18   (*e).taille=1;
19   ((*e).val)[0]=a;
20   for(i=1;i<100;i++) ((*e).val)[i]=0;
21
22   return e;
23 }
24
25 void aff(entierlong *e){
26   int i=(*e).taille-1;
27
28   printf("taille=%d\t",i);
29
30   while(i>=0){
31     if(i==(*e).taille-1) printf("%d",((*e).val)[i]);
32     else{
33       if (((*e).val)[i]<10){printf("00%d", ((*e).val)[i]);}
34       else {
35     if (((*e).val)[i]<100) {printf("0%d", ((*e).val)[i]);}
36     else printf("%d",((*e).val)[i]);
37       }
38     }
39     i--;
40   }
41   printf("\n");
42 }
43
44 /* on suppose a<1000 */
45 entierlong *mult(entierlong *e1, int a) {
46
```

```
47   int i;
48   entierlong *e2 = set(1);
49   (*e2).taille=(*e1).taille;
50
51   for (i=0;i<(*e1).taille;i++)
52     ((*e2).val)[i] = (((*e1).val)[i])*a;
53
54   for(i=0;i<(*e1).taille;i++){
55     ((*e2).val)[i+1] += ((*e2).val)[i] / 1000;
56     ((*e2).val)[i] = ((*e2).val)[i] % 1000;
57     if((i==((*e1).taille)-1) && (((*e2).val[i+1])!=0)) (*e2).taille++;
58   }
59
60   return e2;
61 }
62
63 entierlong *fact(int n){
64   int i;
65   entierlong *e=set(1);
66   entierlong *tmp;
67
68   for (i=2; i<=n; i++){
69     tmp=e;
70     e=mult(e,i);
71     free(tmp);
72   }
73
74   return e;
75 }
76
77 int main(int argc, char *argv[]){
78
79   entierlong *e=set(1);
80
81   e=fact(atoi(argv[1]));
82   aff(e);
83   free(e);
84
```

```
85   return 0;
86 }
```

```
 1 /* Crible d'Eratosthene
 2  ------------------------------------------------ */
 3 #include <stdio.h>
 4 #include <stdlib.h>
 5 #include <math.h>
 6
 7 char *eratosthene(int n) {
 8   int i,j;
 9   char *T;
10   T = malloc((n+1)*sizeof(char));
11   /* T de 0 a n => n+1 case (0 et 1 inutiles) */
12   for (i=0;i<=n;i++) T[i] = 1;
13   for (i=2;i<=n;i++)
14     if (T[i])
15       for (j=2*i; j<=n; j=j+i) T[j] = 0;
16   return T;
17 }
18
19 void affiche(int n) {
20   int i;
21   char *T = eratosthene(n);
22   for (i=2;i<=n;i++)
23     if (T[i]) printf("%d ",i);
24   printf("\n");
25 }
26
27 int main (int argc, char *argv[])
28 {
29   int n = atoi(argv[1]);
30   affiche(n);
31   return 0;
32 }
```