

Documentation of Multiplication and Division

```

void multiply(int a[], int b[], int c[])
/* This is to multiply radix-r multiple precision numbers
a and b, and store the result in c */
{
int i,j;
for (i=0; i<=2*n-1;i++) c[i]=0;
for (i=0; i<=n-1; i++) {
for (j=0; j<= n-1; j++) {
c[i+j+1]=c[i+j+1] + (c[i+j]+a[i]*b[j]) / r;
c[i+j]=(c[i+j] + a[i]*b[j]) % r;
};
};
}

```

This is to multiply a and b, and the product is stored into c. Indices i and j go through multiple precision numbers a and b respectively.

Example

b	5	6	7	j goes on b
a	6	7	8	i goes on a

c	40	48	56	i=0; j=0, 1, 2
		5	6	carry 5, remainder 6
	5	3		

	4	5	3	6
	35	42	49	i=1; j=0, 1, 2
		5	2	
	5	2		
	4	4		

	4	4	2	2
	30	36	42	
		4	4	
	4	4		
	3	8		

	3	8	4	4
		2	6	

```

void divide(int a[], int b[], int c[], int n, int m)
/* This is to divide radix-r multiple precision numbers.
   a is divided by b and the quotient is stored in c.
   The remainder is in a */
{
int i,j,q,x;
int al[100];
  while (b[m-1]==0) m=m-1;
  a[n]=0;
  n=n+1; /* inserted 14 Aug. 2001 */
// The following while loop normalizes b by multiplying it by 2
// We need to multiply a as well to have correct quotient
// At the end a is not the correct remainder due to normalization
  while (b[m-1]<(r / 2)) {
    for(i=0;i<=m-1;i++) b[i]=2*b[i];
    for(i=0;i<=m-1;i++) {
      b[i+1]=b[i+1]+b[i] / r;
      b[i]=b[i] % r;
    }
    for(j=0;j<=n-1;j++) a[j]=2*a[j];
    for(j=0;j<=n-1;j++) {
      a[j+1]=a[j+1]+a[j] / r;
      a[j]=a[j] % r;
    }
  }
  for(j=n-m-1;j>=0;j--) {
    x=r*a[j+m]+a[j+m-1]; //when al[j+i] is negative, d=-al[j+i]deficit
    a[j+m-1]=x; // ..-d....-2r...-r....0....r.....
    a[j+m]=0; // | | | | | | |
    q=x/b[m-1] + 1; //Guess q
    do{
      for(i=m-1;i>=0;i--) al[j+i]=a[j+i];
      /* if q is too large, decrease it by 1 */
      q=q-1;
      count=count+1;
      for(i=m-1;i>=0;i--) al[j+i]=al[j+i]-q*b[i];
      for(i=0;i<=m-2;i++){ // Try to subtract q*b from a
        if(al[j+i]<0){
          al[j+i+1]=al[j+i+1]-(r-1-al[i+j])/r; //borrow from left
          al[j+i]=al[j+i]+((r-1-al[i+j])/r)*r; //remaining at i+j
        }
      } // for i
    } while(al[j+m-1]<0);
    c[j]=q;
    for(i=m-1;i>=0;i--)a[j+i]=al[j+i];
  } // for j
}

```

Until the leading digit of b becomes greater than or equal to $r/2$, we keep multiplying n-digit number a and m-digit number b. Then we guess the first digit for the quotient expressed by q. We copy the relevant portion of a into al, and try to subtract $q*b$. If we hit negative, q is too large, and so subtract 1 from q, and follow the same procedure. If we can subtract, we go to the next position for the quotient. This idea comes from the simple mathematical fact that $a/b = (ax)/(bx)$, where x is a power of 2 in our present problem.

Note that the radix, r, is very large, typically $r = 2^{14}$. Thus good guess of q will lead to vast speed-up.

