Thuật toán Chia để trị : ***Bài toán nhân 2 số nguyên lớn***

Yêu cầu: Cài đặt bài toán nhân 2 số nguyên lớn bằng thuật toán Chia để trị.

Dùng chuỗi ký tự để biểu diễn số nguyên, mỗi số nguyên có n chữ số, n có dạng $n = 2^k$. Đọc dữ liệu từ file, nội dung file nhập có 2 dòng, mỗi dòng biểu diễn một số nguyên, kết thúc bằng ký hiệu xuống dòng.

VD: 1234567887654321
-8765432112345678

```
// Bai toan nhan hai so nguyen lon
// Du lieu cho trong file D://BigInteger.INP
// Giai bang phuong phap CHIA DE TRI
// Dung chuoi ky tu bieu dien cho mot so nguyen
// moi so nguyen co n chu so, n co dang n = 2^k

#include <stdio.h>
#include <string.h>
#include <malloc.h>

typedef char * BigInteger;

void ReadFromFile(BigInteger x, BigInteger y){
    FILE *f;
    f=fopen("BigInt.txt", "r");
        fgets(x,255,f);
        x[strlen(x)-1]='\0';
```

```
        fgets(y,255,f);
        y[strlen(y)-1]='\0';
    fclose(f);
}
int Sign(BigInteger x){
        return (x[0]=='-' ? -1 : 1);
}

BigInteger Right(BigInteger x, int n){
        int i,l = strlen(x);
        BigInteger ptr=x+l-1;
        for (i=l-1; i>l-n ; i--) ptr--;
        return ptr;
}

BigInteger Left(BigInteger x, int n){
        int i;
        BigInteger L;
        L=(char*) malloc(sizeof(char)*256);
        for(i=0;i<n;i++) L[i]=x[i];
        L[n]='\0';
        return L;
}
```

```
BigInteger ABS(BigInteger x){
        if(Sign(x)==-1)
          return(Right(x,strlen(x)-1));
        else return x;
}

BigInteger Nhan10_mu_n (BigInteger x, int n){
        int i;
        BigInteger temp;
        temp=(char*)malloc(sizeof(char)*256);
        strcpy(temp,x);
        int l=strlen(temp);
        for(i=0;i<n;i++) temp[l+i]='0';
        temp[l+n]='\0';
        return temp;
}

BigInteger Reverse(BigInteger n){
  BigInteger kq;
  kq = (char*) malloc(sizeof(char)*256);
  int L = strlen(n);
  int i;
  for(i=0; i<L; i++)
        kq[i]=n[L-i-1];
```

```
   kq[L]='\0';
   return kq;
}

int Zero(BigInteger n){
    return n[0]== '0';
}

int Positive(BigInteger n){
    return n[0]>'0';
}

int Negative(BigInteger n){
    return n[0]=='-' && !Zero(n);
}

int Not_Negative(BigInteger n){
    return Zero(n) || Positive(n);
}

//
int Not_Positive(BigInteger n){
    return Zero(n) || Negative(n);
}
```

```
// Ham xet xem 2 so co bang nhau hay kh
int Equal(BigInteger n, BigInteger m){
   return !strcmp(n,m);
}



/*  Ham xet xem so n co nho hon so m
 Ta xet cac truong hop sau
 0- neu n bang m   => Khong nho hon
 1- n am va m khong am => n<m
 2- n bang khong va m duong       => n<m
 3- n khong am va m am            => n>m
 4- n duong va m khong duong      => n>m
 5- n va m cung duong va do dai cua n nho hon m => n<m
 6- n va m cung khong am, cung do dai, xet tung ky tu cho den khi gap n[i]<m[i]
         thi n<m
 7- n va m cung am, thi n<m khi abs(m)<abs(n)
 */


 int Less_Than(BigInteger n, BigInteger m){
    if (Equal(n,m))
       return 0;
    if (Negative(n)&& Not_Negative(m))
```

```
        return 1;
    if (Zero(n)&& Positive(m))
        return 1;
    if (Not_Negative(n)&& Negative(m))
        return 0;
    if (Positive(n)&& Not_Positive(m))
        return 0;
    if (Not_Negative(n)&& Not_Negative(m))
        if (strlen(n)!=strlen(m))
            return strlen(n)<strlen(m);
        else {
             int i=0;
             while (n[i]==m[i]) i++;
             return (n[i]<m[i]);
             }
    if (Negative(n)&& Negative(m))
            return Less_Than(ABS(m),ABS(n));
}
// Xet xem so n co lon hon so m hay khong

int Greater_Than(BigInteger n, BigInteger m){
   return Less_Than(m,n);
}
```

```
int Less_Or_Equal(BigInteger n, BigInteger m){
    return Less_Than(n,m) || Equal(n,m);
}

int Greater_Or_Equal(BigInteger n, BigInteger m){
    return Greater_Than(n,m) || Equal(n,m);
}

// Ham tru so nguyen n1 cho n2 voi gia thiet n1>=n2

BigInteger Subtract1(BigInteger x, BigInteger y){
   BigInteger kq,n,m;
   kq = (char*) calloc(256,sizeof(char));
   n = (char*) calloc(256,sizeof(char));
   m = (char*) calloc(256,sizeof(char));
   n = Reverse(x);
   m = Reverse(y);
   int L1=strlen(n);
   int L2=strlen(m);
   int i, nho=0;
   for (i=0; i<L2; i++)
      if (n[i] >= m[i] + nho) {
         kq[i]=(n[i]-m[i]-nho)+48;
         nho=0;
```

```
      }else {
        kq[i]=(n[i]+10-m[i]-nho)+48;
        nho=1;
       }
    if (nho==0)
     for (i=L2; i<L1; i++) kq[i]=n[i];
    else
     for (i=L2; i<L1; i++)
     if (n[i]-48 >= nho) {
      kq[i]=(n[i]-nho);
      nho=0;}
     else{
      kq[i]=(n[i]+10-nho);
      nho=1;
     }
  kq[strlen(kq)]='\0';
  return Reverse(kq);
}

// nhan mot so nguyen voi so 1 hoac -1
BigInteger MultS(BigInteger x, int s){
        if(s==1) return x;
        else {
                int i,l=strlen(x);
```

```
                BigInteger temp;
            temp=(char*)malloc(sizeof(char)*256);
        temp[0]='-';
        for(i=1;i<=l;i++) temp[i]=x[i-1];
        temp[l+1]='\0';
        return temp;
    }
}

BigInteger Subtract(BigInteger x, BigInteger y){
          if (Greater_Or_Equal(x,y))
            return Subtract1(x,y);
          else
            return MultS(Subtract1(y,x), -1);
}



// cong 2 so nguyen khong am

BigInteger Add1(BigInteger n1, BigInteger n2){
   BigInteger kq,n,m;
   kq = (char*) calloc(256,sizeof(char));
   n = (char*) calloc(256,sizeof(char));
   m = (char*) calloc(256,sizeof(char));
```

```
strcpy(n,Reverse(n1));
strcpy(m, Reverse(n2));
int L1=strlen(n);
int L2=strlen(m);
int i, L, H, nho=0;
if (L1>=L2){
 H=L1;
 L=L2;
}else {
 H=L2;
 L=L1;
}
for (i=0; i<L; i++){
    kq[i]=(n[i]+m[i]-96+nho)%10+48;
    nho = (n[i]-48+m[i]-48+nho)/10;
}

if (L1>=L2)
  for (i=L; i<H; i++){
    kq[i]=(n[i]-48+nho)%10+48;
    nho = (n[i]-48+nho)/10;
 }
else
  for (i=L; i<H; i++){
```

```
      kq[i]=(m[i]-48+nho)%10+48;
      nho = (m[i]-48+nho)/10;
    }
  if (nho>0) strcat(kq,"1");
  kq[strlen(kq)]='\0';
  return (Reverse(kq));
}

// Cong hai so bat ky

BigInteger Add(BigInteger n1, BigInteger n2){
        if (Not_Negative (n1))
          if (Not_Negative (n2)) return Add1(n1,n2);
          else return Subtract(n1,ABS(n2));
        else
          if (Not_Negative (n2))return Subtract(n2,ABS(n1));
          else return MultS(Add1(ABS(n1),ABS(n2)),-1);
}

// Cong 3 so nguyen

BigInteger Add3(BigInteger n1, BigInteger n2, BigInteger n3){
        return Add(Add(n1,n2),n3);
}
```

```
// Nhan 2 so nguyen co mot chu so

BigInteger Mult1(BigInteger x, BigInteger y){
        BigInteger Temp;
        Temp=(char*)malloc(sizeof(char)*3);
        int nho;
    Temp[0] = (x[0]-48)*(y[0]-48)%10+48;
        nho = (x[0]-48)*(y[0]-48)/10;

        if (nho>0){
     Temp[1]=nho+48;
     Temp[2]='\0';
        }
        else
     Temp[1]='\0';
        return Reverse(Temp);
}

BigInteger Mult(BigInteger X, BigInteger Y, int n){
   BigInteger m1,m2,m3,A,B,C,D;
   int s; // Luu tru dau cua tich XY
   s = Sign(X)*Sign(Y);
```

```
  X = ABS(X); //Lay tri tuyet doi cua X
  Y = ABS(Y);

  if (n == 1)  return MultS(Mult1(X,Y),s);

  A = Left(X, n/2);
  B = Right(X, n/2);
  C = Left(Y, n/2);
  D = Right(Y, n/2);
  m1 = Mult(A,C, n/2);
  m2 = Mult(Subtract(A,B),Subtract(D,C), n/2);
  m3 = Mult(B,D, n/2);

  return
      MultS(Add3(Nhan10_mu_n(m1,n),Nhan10_mu_n(Add3(m1,m2,m3),n/2)
      , m3),s);
}

int main(){
   BigInteger x, y;
      x=(char*)malloc(sizeof(char)*256);
      y=(char*)malloc(sizeof(char)*256);
      ReadFromFile(x,y);
      printf("\nSo nguyen X= %s\n\n",x);
      printf("So nguyen Y= %s\n\n",y);
```

```
        printf("Tich So XY= %s\n",Mult(x,y,strlen(ABS(x))));
        free(x);
        free(y);
        return 0;
}
```