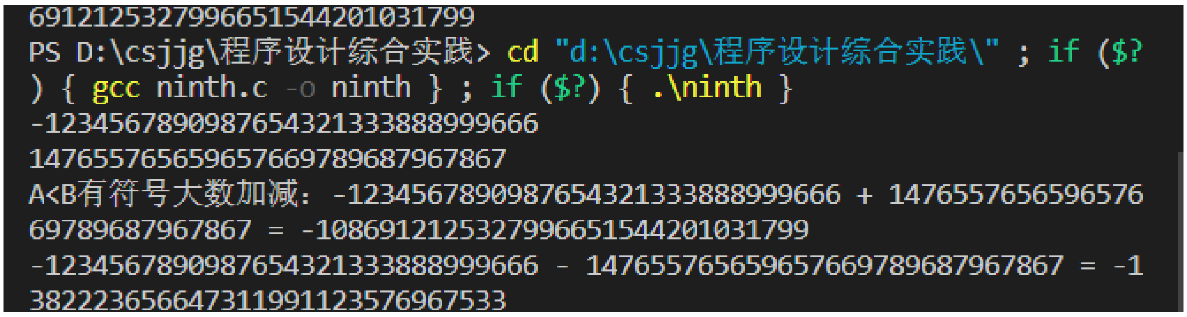
```c
#include <stdio.h>
#include <malloc.h>
#include <assert.h>

struct  Node
{
    int digit;
    struct Node *next, *prev;
};
struct UBigNumber
{
    int     digitCount;
    struct Node *pHead, *pTail;
};

struct UBigNumber InputUBN ();
void PrintUBN (struct UBigNumber ubn);
struct UBigNumber AddUBN (struct UBigNumber *pA, struct UBigNumber *pB);
void DestoryUBN (struct UBigNumber *pA);
void _InitUBN (struct UBigNumber *pUBN);
void _AppendDigit (struct UBigNumber *pUBN, int digit);
void _AppendFrontDigit (struct UBigNumber *pUBN, int digit);
void _Normalize (struct UBigNumber *pUBN);
struct Node *_NewNode ();
struct UBigNumber MinusUBN(struct UBigNumber *pA, struct UBigNumber *pB);
int Compare(struct UBigNumber *pA, struct UBigNumber *pB);//1代表前面大，-1代表后面大
struct UBigNumber MinusBN (struct UBigNumber *pA, struct UBigNumber *pB);//有符号大数减
struct UBigNumber AddBN (struct UBigNumber *pA, struct UBigNumber *pB);//有符号大数加



int main ()
{   struct UBigNumber A, B, C,D;
    A = InputUBN ();
    B = InputUBN ();
/*    C = AddUBN (&A, &B);
    D = MinusUBN(&A, &B);

    PrintUBN (A);
    printf (" + ");
    PrintUBN (B);
    printf (" = ");
    PrintUBN (C);
    printf("\n");
    PrintUBN (A);
    printf (" - ");
    PrintUBN (B);
    printf (" = ");
    PrintUBN (D);   */
    int com = Compare(&A,&B);
    if(com==1){
        printf("A>B");
    }else if(com==-1){
        printf("A<B");
    }else{
        printf("A=B");
    }
    printf("有符号大数加减：");
    PrintUBN (A);
    printf (" + ");
    PrintUBN (B);
    printf (" = ");
    PrintUBN (AddBN(&A,&B));
    printf("\n");
    PrintUBN (A);
    printf (" - ");
    PrintUBN (B);
    printf (" = ");
    PrintUBN (MinusBN(&A,&B));
    DestoryUBN (&A);
    DestoryUBN (&B);
    //DestoryUBN (&C);
    return 0;
}
int Compare(struct UBigNumber *pA, struct UBigNumber *pB){
    if(pA->pHead->digit==1 && pB->pHead->digit==0){//前 负，后 正
         return -1;//后面大
    }else if(pA->pHead->digit==0 && pB->pHead->digit==1){
        return 1;//前面大
    }else if(pA->pHead->digit==0 && pB->pHead->digit==0){
        //先比较位数，多者 大
        int a=0,b=0;
        struct Node *p1,*p2;
        p1=pA->pHead->next;
        p2=pB->pHead->next;
        while(p1!=pA->pTail){
          a++;
          p1=p1->next;
        }
        a++;
        while(p2!=pB->pTail){
          b++;
          p2=p2->next;
        }
        b++;
        if(a>b){
            return 1;
        }else if(a<b){
            return -1;
        }else{//位数相等时，从头开始比，先大者大
            p1=pA->pHead->next;
            p2=pB->pHead->next;
            while(p1!=pA->pTail && p2!=pB->pTail){
                if(p1->digit > p2->digit){
                    return 1;
                }else if(p1->digit < p2->digit){
                    return -1;
                }else {
                    p1=p1->next;
                    p2=p2->next;

                }
            }
            return 0;//两者相等
        }
    }else{//都为负的情况，为都为正的情况符合相反,1改为-1，-1改为1
        //先比较位数，多者 小
        int a=0,b=0;
        struct Node *p1,*p2;
        p1=pA->pHead->next;
        p2=pB->pHead->next;
        while(p1!=pA->pTail){
          a++;
          p1=p1->next;
        }
        a++;
        while(p2!=pB->pTail){
          b++;
          p2=p2->next;
        }
        b++;
        if(a>b){
            return -1;
        }else if(a<b){
            return 1;
        }else{//位数相等时，从头开始比，先大者小
            p1=pA->pHead->next;
            p2=pB->pHead->next;
            while(p1!=pA->pTail && p2!=pB->pTail){
                if(p1->digit > p2->digit){
                    return -1;
                }else if(p1->digit < p2->digit){
                    return 1;
                }else {
                    p1=p1->next;
                    p2=p2->next;

                }
            }
            return 0;//两者相等
    }
}
}

struct UBigNumber InputUBN ()
{
    struct UBigNumber result;
    _InitUBN(&result);

    char ch;
    do
        ch = getchar ();
    while ((ch < '0' || ch > '9') && ch!='-');
    if(ch=='-'){
        result.pHead->digit=1;//1代表是负数
        ch = getchar ();
    }
    while (ch >= '0' && ch <= '9')
```

```c
    {
        _AppendDigit (&result, ch - '0');
        ch = getchar ();
    }
    _Normalize(&result);
    return result;
}
void PrintUBN (struct UBigNumber ubn)
{
    assert (ubn.digitCount > 0 && ubn.pHead->next != NULL);
    struct Node *la = ubn.pHead->next;
    if(ubn.pHead->digit==1){
        printf("-");
    }
    while (la)
    {
        printf ("%d", la->digit);
        la = la->next;
    }
}
struct UBigNumber MinusUBN (struct UBigNumber *pA, struct UBigNumber *pB)
{
    struct UBigNumber result, *pResult = &result;
    int flag=0;
    _InitUBN(pResult);
    struct Node *p1, *p2;
    p1 = pA->pTail;
    p2 = pB->pTail;
    while (p1 != pA->pHead && p2 != pB->pHead)
    {
        int digit = p1->digit - p2->digit +flag;
        flag=0;
        if(digit>=0){
        _AppendFrontDigit (pResult, digit);
        p1 = p1->prev;
        p2 = p2->prev;
        }else{
        _AppendFrontDigit (pResult, digit+10);
        p1 = p1->prev;
        p2 = p2->prev;
        flag=-1;
        }
    }
    while (p1 != pA->pHead->next && p1 != pA->pHead)
    {
        int digit = (p1->digit) + flag;
        flag=0;
        if(digit<0){
        _AppendFrontDigit (pResult, digit+10);
        p1 = p1->prev;
        flag=-1;;
        }else{
        _AppendFrontDigit (pResult, digit);
        p1 = p1->prev;
        }
    }
    if((p1->digit +flag)>0 && p1 == pA->pHead->next){
        _AppendFrontDigit (pResult, p1->digit+flag);
    }else{

    }

    return result;
}
struct UBigNumber MinusBN (struct UBigNumber *pA, struct UBigNumber *pB)//有符号大数减
{
    if(pA->pHead->digit==0 && pB->pHead->digit==0){
        int flag=Compare(pA,pB);
        if(flag==1 || flag==0){
          return MinusUBN(pA,pB);
        }else{
            struct UBigNumber u = MinusUBN(pB,pA);
            u.pHead->digit=1;
            return u;
        }
    }else if(pA->pHead->digit==1 && pB->pHead->digit==1){
        int flag=Compare(pA,pB);
        if(flag==1 || flag==0){//前者大，后者小，相当于后减前，加负号
          return MinusUBN(pB,pA);
        }else{
            struct UBigNumber u = MinusUBN(pA,pB);//前减后，加负号，前小说明负的多
            u.pHead->digit=1;
            return u;
        }

    }else if(pA->pHead->digit==1 && pB->pHead->digit==0){
            struct UBigNumber u = AddUBN(pA,pB);
            u.pHead->digit=1;
            return u;
    }else if(pA->pHead->digit==0 && pB->pHead->digit==1){
        return AddUBN(pA,pB);
    }

}
struct UBigNumber AddBN (struct UBigNumber *pA, struct UBigNumber *pB)//有符号大数加
   {
     if(pA->pHead->digit==0 && pB->pHead->digit==0){
        return AddUBN(pA,pB);
     }else if(pA->pHead->digit==1 && pB->pHead->digit==1){
        struct UBigNumber u = AddUBN(pA,pB);
        u.pHead->digit=1;
        return u;
     }else if(pA->pHead->digit==0 && pB->pHead->digit==1){//b为负的
         pB->pHead->digit=0;
         int flag = Compare(pA,pB);
         pB->pHead->digit=1;
         if(flag==1 || flag ==0){
             return MinusUBN(pA,pB);
         }else if(flag == -1){
             struct UBigNumber u = MinusUBN(pB,pA);
             u.pHead->digit=1;
             return u;
         }
```

```c

     }else {//a为负的
         pA->pHead->digit=0;
         int flag = Compare(pA,pB);
         pA->pHead->digit=1;
         if(flag==-1 || flag ==0){
             return MinusUBN(pB,pA);
         }else if(flag == 1){
             struct UBigNumber u = MinusUBN(pA,pB);
             u.pHead->digit=1;
             return u;
         }
     }
   }
struct UBigNumber AddUBN (struct UBigNumber *pA, struct UBigNumber *pB)
{
    struct UBigNumber result, *pResult = &result;
    _InitUBN(pResult);
    int iCarry = 0;
    struct Node *p1, *p2;
    p1 = pA->pTail;
    p2 = pB->pTail;
    while (p1 != pA->pHead && p2 != pB->pHead)
    {
        int digit = p1->digit + p2->digit + iCarry;
        iCarry = digit / 10;
        digit %= 10;
        _AppendFrontDigit (pResult, digit);
        p1 = p1->prev;
        p2 = p2->prev;
    }
    while (p1 != pA->pHead)
    {
        int digit = p1->digit + iCarry;
        iCarry = digit / 10;
        digit %= 10;
        _AppendFrontDigit (pResult, digit);
        p1 = p1->prev;
    }
        while (p2 != pB->pHead)
    {
        int digit = p2->digit + iCarry;
        iCarry = digit / 10;
        digit %= 10;
        _AppendFrontDigit (pResult, digit);
        p2 = p2->prev;
    }
    if (iCarry != 0)
        _AppendFrontDigit (pResult, iCarry);
    return result;
}
void DestoryUBN (struct UBigNumber *pUBN)
{
    while (pUBN->pHead != NULL)
    {
        struct Node *p =  pUBN->pHead;
        pUBN->pHead = p->next;
        free (p);
    }
}
void _InitUBN (struct UBigNumber *pUBN)
{
    struct Node *p = _NewNode ();
    p->digit=0;//代表是正的，后面输入符号会修改
    pUBN->pHead = pUBN->pTail = p;
    p->next = p->prev = NULL;
    pUBN->digitCount = 0;
    }
void _AppendDigit (struct UBigNumber *pUBN, int digit)
{
    if (pUBN->digitCount == 1 && pUBN->pTail->digit == 0)
    {//直到出现非0数字才可以结束
        pUBN->pTail->digit = digit;
        return;
    }
    struct Node *p = _NewNode ();  //数字链表添加一个结点
    p->digit = digit;
    p->next = NULL;
    p->prev = pUBN->pTail;
    pUBN->pTail->next = p;
    pUBN->pTail = p;
    ++pUBN->digitCount;
}
void _AppendFrontDigit (struct UBigNumber *pUBN, int digit)
{
    struct Node *p = _NewNode ();
    p->digit = digit;
    p->next = pUBN->pHead->next;
    if (p->next != NULL)
        p->next->prev = p;
    p->prev = pUBN->pHead;
    pUBN->pHead->next = p;
    if (pUBN->pTail == pUBN->pHead)
        pUBN->pTail = p;
    ++pUBN->digitCount;
}
void _Normalize (struct UBigNumber *pUBN)
{
    if (pUBN->digitCount == 0)
        _AppendDigit (pUBN, 0);
    while (pUBN->digitCount > 1 && pUBN->pHead->next->digit == 0)
    {//过滤0
        struct Node *p;
        p = pUBN->pHead->next;
        pUBN->pHead->next = p->next;
        p->next->prev = pUBN->pHead;
        free (p);
        --pUBN->digitCount;
    }
}

struct Node *_NewNode ()
{
    struct Node *p;
    p = (struct Node *) malloc (sizeof (struct Node));
    if (p == NULL)
    {
        printf ("Error : out of memory\n");
        exit (-1);
    }
    return p;
}
```



```
6912125327996651544201031799
PS D:\csjjg\程序设计综合实践> cd "d:\csjjg\程序设计综合实践\" ; if ($?
) { gcc ninth.c -o ninth } ; if ($?) { .\ninth }
-12345678909876543213338889996666
14765576565965766978968796786 7
A<B有符号大数加减: -12345678909876543213338889996666 + 1476557656596576
69789687967867 = -10869121253279966515442010317 99
-12345678909876543213338889996666 - 14765576565965766978968796786 7 = -1
3822236566473119911235769675 33
```