ICS BUGLAB REPORT

胡译文 2021201719

最新版，补了一些过程。但因为很多都是以前学竞赛时遇到过的bug，已经作为经验储存在脑子里所以只要一读就能发现问题）

SOLUTIONS

shuffle

经典的使用异或交换错误（老师有讲）和变量名写错（仔细读代码）。

```
shuffle.cpp
--- buggy/shuffle.cpp   2022-10-14 10:51:25.000000000 +0800
+++ fixed/shuffle.cpp   2022-10-18 08:15:08.000000000 +0800
@@ -3,0 +4,3 @@ void swap(int & a, int & b) {
+    if (a == b) {
+        return ;
+    }
@@ -24 +27 @@ int main() {
-        if(a < 0 || a >= n || b < 0 || b >= n) {
+        if(a < 0 || a >= m || b < 0 || b >= m) {

```

polycalc

经典的没有初始化（仔细读代码）和T了就知道的快速幂（线性算法优化为对数）。传参如果改成引用会更好，但非必要不改动。

```
polycalc.cpp
--- buggy/polycalc.cpp  2022-10-14 10:51:25.000000000 +0800
+++ fixed/polycalc.cpp  2022-10-18 08:36:17.000000000 +0800
@@ -35 +35 @@ ElemTypeB calc() {
-     ElemTypeB result;
+     ElemTypeB result(0);
@@ -45,3 +45,8 @@ ElemTypeB calc() {
-     ElemTypeB result;
-     for(int i = 0; i <= node.exp - 1; i++) {
-         result *= node.base;
+     ElemTypeB result(1), multiplier = node.base;
+     ElemTypeA exp = node.exp;
+     while (exp) {
+         if (exp & 1) {
+             result *= multiplier;
+         }
+         exp >>= 1;
+         multiplier *= multiplier;
```

violetStore

“所有bug”和“非必要改动”比较难平衡，本题又特别强调了所有bug。排一下错误等级，可以从后往前删:)

- 结果错误：`malloc` 不会调用构造函数，要用 `new` ~~或 `make_unique` （要引入库）~~（C++知识）
- 结果错误：`n` 没有初始化（仔细读代码）
- 内存管理错误：数组要用 `delete[]` （C++知识）
- 内存管理错误：数组地址不能 `++` （C++知识）
- 内存管理错误：`free` 不会调用析构函数，直接删掉离开作用域自动析构就行（C++知识）
- 鲁棒性：求最小值直接硬编码（经验）
- 鲁棒性：添加超过 3 个物品（经验）

- 潜在bug：宏定义没加括号导致优先级错误（~~std::min不是更好吗~~）（以前看到过）

```
violetStore.cpp
--- buggy/violetStore.cpp   2022-10-14 10:51:25.000000000 +0800
+++ fixed/violetStore.cpp   2022-10-18 08:50:07.000000000 +0800
@@ -3 +3 @@
-#define min(a,b) a<=b?a:b
+#define min(a,b) (a)<=(b)?(a):(b)
@@ -13,0 +14 @@ public:
+        : n(0)
@@ -21,2 +22,2 @@ public:
-        delete items;
-        delete prices;
+        delete[] items;
+        delete[] prices;
@@ -26,2 +27,4 @@ public:
-        *items++ = name;
-        *prices++ = price;
+        if (n >= 3) return;
+        *(items + n) = name;
+        *(prices + n) = price;
+        n++;
@@ -37 +40,4 @@ public:
-        return min(min(prices[0], prices[1]), prices[2]);
+        int min_price = 1e9;
+        for (int i = 0; i < n; ++i)
+            min_price = min(min_price, prices[i]);
+        return min_price;
@@ -43 +49 @@ int main()
-    price* test = (price*)malloc(sizeof(price));
+    price* test = new price();
@@ -49 +54,0 @@ int main()
-    free(test);

```

宏定义还有更完美的版本：（但没必要略）

```
#define min(a, b) ({     \
    typeof(x) _a = (a); \
    typeof(y) _b = (b); \
    (void) (&_a == &_b);\
    _a < _b ? _a : _b; })
})
```

swapCase

经典T，改快读（getchar效率很高）。可以加一个对 `strlen` 的限制，但既然是做题就没必要。

```
swapCase.cpp
--- buggy/swapCase.cpp  2022-10-14 10:51:25.000000000 +0800
+++ fixed/swapCase.cpp  2022-10-18 09:00:36.000000000 +0800
@@ -8,2 +8,6 @@ int main(){
-    scanf("%s", s);
-    for(int i = 0; i < strlen(s); ++i){
+    int strlen = 0;
+    char ch;
+    while((ch = getchar()) != '\n') {
+        s[strlen++] = ch;
+    }
+    for(int i = 0; i < strlen; ++i){

```

xorsum

经典没初始化（读代码）和快读顺序问题（逗号运算符顺序从右到左）。可以~~重载逗号运算符或~~分开来写。

```
xorsum.cpp
--- buggy/xorsum.cpp    2022-10-14 10:51:25.000000000 +0800
+++ fixed/xorsum.cpp    2022-10-18 09:10:20.000000000 +0800
@@ -5 +5 @@ int q;
```

```
-int ans;
+int ans=0;
@@ -31,2 +31,5 @@ int main(){
-    for(int i=0;i<q;i++)
-        Replace(ReadInt(), ReadInt());
+    for(int i=0;i<q;i++) {
+        int pos = ReadInt();
+        int value = ReadInt();
+        Replace(pos, value);
+    }

```

mergeIntervals

首先一个不那么常见的错误，比较函数应该返回严格 `Order` （仔细读代码可以发现）。其次因为是对左端点的偏序，右端点不一定是顺序的，取最大值即可（逻辑错误，推一遍就知道了）。

```
mergeIntervals.cpp
--- buggy/mergeIntervals.cpp    2022-10-14 10:51:24.000000000 +0800
+++ fixed/mergeIntervals.cpp    2022-10-18 09:22:43.000000000 +0800
@@ -9 +9 @@ bool compare(const Range& x, const Range
-    return x.l <= y.l;
+    return x.l < y.l;
@@ -26 +26 @@ int main(){
-            last.r = it->r;
+            last.r = std::max(it->r, last.r);

```

8num

经典的内存管理错误。`curState` 申明方式改成 `new` 并修复一下 `delete` 就行（C++知识）。析构函数理论上最好补上一个递归析构 `parent` 预防不细心，但因为已经有循环析构就不改了。虽然说这里的 `new` 和下面的 `malloc` 非常不统一，但是为了遵循“非必要不改动”，就不改了。

```
8num.cpp
--- buggy/8num.cpp  2022-10-18 19:47:01.000000000 +0800
+++ fixed/8num.cpp  2022-10-18 20:33:59.000000000 +0800
@@ -87,2 +87 @@ int IDS(int max_depth){
-        State curs = State(que.top().first);
-        State* curState = &curs;
+        State* curState = new State(que.top().first);
@@ -94,0 +94 @@ int IDS(int max_depth){
+            State* tmp;
@@ -96,0 +97 @@ int IDS(int max_depth){
+                tmp = curState;
@@ -98 +99 @@ int IDS(int max_depth){
-                free(curState);
+                delete tmp;

```

segtree

经典的 `long long` 错误，题目专门强调 $\leq 10^9$ 就知道肯定会溢出。值得注意的是这里 `sum` 表面没有初始化，但因为会从子节点或叶节点更新，所以其实是初始化了的。`lch` 最好也是初始化一下（读代码），虽然只有在析构的时候可能有问题。

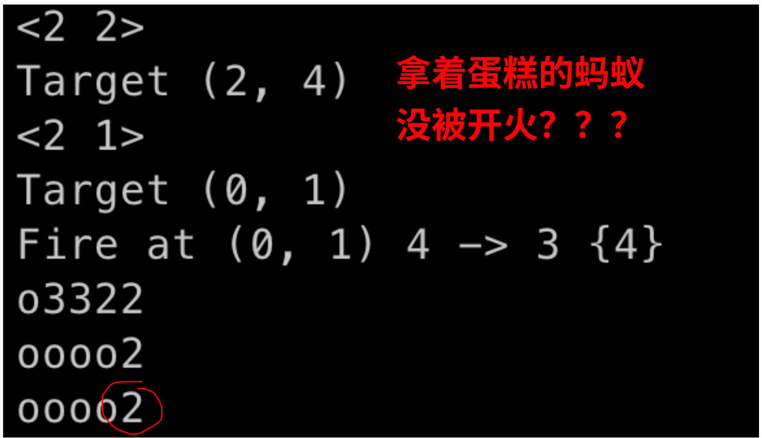
```
segtree.cpp
--- buggy/segtree.cpp   2022-10-14 10:51:25.000000000 +0800
+++ fixed/segtree.cpp   2022-10-21 23:45:05.000000000 +0800
@@ -13 +13 @@ struct Node{
-   int sum;
+   long long sum;
@@ -21 +21 @@ struct Node{
-   int Query(const int&,const int&);
```

```
+   long long Query(const int&,const int&);
@@ -42 +42 @@ int main(){
-           printf("%d\n",N->Query(l,r));
+           printf("%lld\n",N->Query(l,r));
@@ -54 +54 @@ Node::Node(int l,int r){
-   if(l==r)
+   if(l==r){
@@ -55,0 +56,3 @@ Node::Node(int l,int r){
+                this->lch=nullptr;
+                this->rch=nullptr;
+        }
@@ -86 +89 @@ void Node::Add(const int& l,const int& r
-int Node::Query(const int& l,const int& r){
+long long Node::Query(const int& l,const int& r){

```

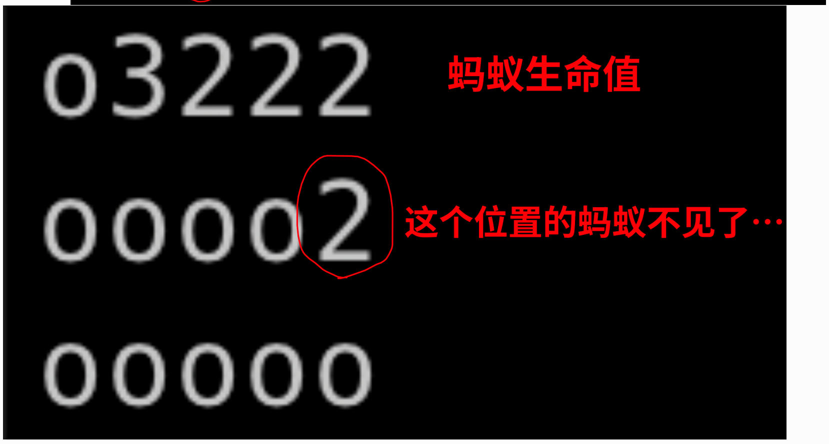
antbuster

好题！当我看到我的蚂蚁消失的时候我的内心是震惊的……然后还有拿着蛋糕的蚂蚁不会被开火等等问题……



```
<2 2>
Target (2, 4)
<2 1>
Target (0, 1)
Fire at (0, 1) 4 -> 3 {4}
o3322
oooo2
oooo2
```

拿着蛋糕的蚂蚁
没被开火？？？



```
o3222
oooo2
ooooo
```

蚂蚁生命值

这个位置的蚂蚁不见了…

详细的修改过程就不赘述，列举一些我以为是错误而多改的地方：

- 因为地图范围是左闭右闭，地图大小是 $(n+1)\times(m+1)$
- x 轴和 y 轴与笛卡儿坐标系是相反的，因此代码中没错
- 虽然 `CheckAvailable` 函数名相同，在结构体内调用遮盖掉全局的函数，但在本题的用法是正确的
- 找 `CakeCarrier` 时找到了可以 `break` 但也没啥必要
- 代码全黏成一坨看着眼睛疼，下次建议正常一点
- 判断 `Cross` 理论上切点（大于等于号取等）也算，但非必要就不修改了

而实际需要修改的地方：

- 初始化（出现率极高的bug，仔细读代码就行）
- 对负数取模（我猜可能不信但是真的是经验）
- 直线应该用标准式（会产生除0错误）
- 信息素为负（我把地图信息都打印出来了，很明显不应该是负的）

```
antbuster.cpp
--- buggy/antbuster.cpp 2022-10-14 10:51:24.000000000 +0800
+++ fixed/antbuster.cpp 2022-10-21 23:00:34.000000000 +0800
@@ -35,3 +35,3 @@ int t;
-int clk;  // Global clock
-int spn;  // Ant spawn count
-bool END;
+int clk=0;  // Global clock
+int spn=0;  // Ant spawn count
+bool END=false;
@@ -39 +39 @@ int s,d,r;
-Ant* cakeCarrier;
+Ant* cakeCarrier=NULL;
@@ -80 +80,2 @@ void OneSecond(){
-                i->HP=std::min(i->mxHP,i->HP+i->mxHP/2);
+                i->HP=std::min(floor(i->mxHP),i->HP+(i->mxHP/2.));
+                break;
@@ -115,2 +116 @@ void Tower::Fire(){
-    double k=dy/dx;
-    double b=this->y-k*this->x;
+    double b=dx*this->y-dy*this->x;
@@ -118 +118 @@ void Tower::Fire(){
-        if(Cross(k,-1.0,b,i->x,i->y)&&InSegment(this->x,this->y,target->x,target->y,i->x,i->y)&&SqrEucDis(this->x,this->y,i->x,i->y)<=SqrEucDis(this->x,this->y,target->x,target->y)){
```

```
+        if(Cross(dy,-dx,b,i->x,i->y)&&InSegment(this->x,this->y,target->x,target->y,i->x,i->y)&&SqrEucDis(this->x,this->y,i->x,i->y)<=SqrEucDis(this->x,this->y,target->x,target->y)){
@@ -152 +151,0 @@ void Ant::NormalMove(int dir){
-        return;
@@ -165 +164 @@ void Ant::SpecialMove(int dir){
-    dir=(dir-1)%4;
+    dir=(dir+3)%4;
@@ -167 +166 @@ void Ant::SpecialMove(int dir){
-        dir=(dir-1)%4;
+        dir=(dir+3)%4;
@@ -199 +198 @@ void DecreaseSignal(){
-            --sign[i][j];
+            sign[i][j] -= !!(sign[i][j]);

```

softDuble

第一眼看上去 `1` 右移会溢出，全部替换成 `1ull` 。这是允许C++极大的自由度允许隐式类型转换带来的取舍，希望不是“非必要改动”。~~然后就找不到错误了。~~

类似的错误是左移 `ediff` 时大于 64 会溢出。（右移有问题当然就检查了一遍左移，）

然后开始构造测试点处理特殊样例。【+1、-1、+0、-0、+999...、-999...】+【加、减、乘、除】+【+1、-1、+0、-0、+999...、-999...】

发现以下问题：

- 除法 `inf` 符号（构造样例）
- `0.+(-0.)` 输出 `-0` （这个看起来数据和真实C++程序并不一样，调了好久。。。）

最后就是读代码（肯定先读重点部分）。。。datalab还是有点用，对于浮点数四舍六入五成双有了较为深刻的理解。用 `*` 表示要舍弃的部分， `#=^` 等都表示要保留的部分，其中最后两个分别为倒数第二和倒数第一位 。那么“入”的条件有两个，第一是 `^` 为 `1` （五或六），第二个是 `=` 或 `*` 包含一个 `1` （五成双或六）。

```
0000011100
###=^*****
```

代码：

```
softDouble.cpp
--- buggy/softDouble.cpp    2022-10-14 10:51:25.000000000 +0800
+++ fixed/softDouble.cpp    2022-10-21 23:11:24.000000000 +0800
@@ -5,3 +5,3 @@ const int BUFFER_LEN = 100010;
-const uint64_t INF = 0x7FF0000000000000;
-const uint64_t NaN = 0x7FF00000001F1E33;
-const uint64_t NINF = 0xFFF0000000000000;
+const uint64_t INF = 0x7FF0000000000000ull;
+const uint64_t NaN = 0x7FF00000001F1E33ull;
+const uint64_t NINF = 0xFFF0000000000000ull;
@@ -58 +58 @@ int main(){ // Function: Parse & Evaluat
-                    assert(opstop != stackops);
+                    assert(opstop != stackops);  // not empty
@@ -125 +125 @@ inline bool isNaN(uint64_t x){
-    return (Exp(x) == (1 << 11) - 1) && (Fraction(x) & ((1 << 52) - 1)) != 0;
+    return (Exp(x) == (1ull << 11) - 1ull) && (Fraction(x) & ((1ull << 52) - 1ull)) != 0;
@@ -129 +129 @@ inline bool isINF(uint64_t x){
-    return (Exp(x) == (1 << 11) - 1) && (Fraction(x) & ((1 << 52) - 1)) == 0;
+    return (Exp(x) == (1ull << 11) - 1ull) && (Fraction(x) & ((1ull << 52) - 1ull)) == 0;
@@ -133 +133 @@ inline bool isZero(uint64_t x){
-    return (x & ((1 << 63) - 1)) == 0;
+    return (x & ((1ull << 63) - 1ull)) == 0;
@@ -166 +166 @@ uint64_t add(uint64_t lhs, uint64_t rhs)
-        uint64_t cur = LowBit(rhsf) >> ediff;
+        uint64_t cur = (ediff < 64) ? (LowBit(rhsf) >> ediff) : 0;
@@ -175 +175 @@ uint64_t add(uint64_t lhs, uint64_t rhs)
-    while(ansf >= (1 << 54)){
+    while(ansf >= (1ull << 54)){
@@ -183 +183 @@ uint64_t add(uint64_t lhs, uint64_t rhs)
```

```
-        assert(ansexp < (1 << 53));
+        assert(ansexp < (1ull << 53));
@@ -190 +190 @@ uint64_t add(uint64_t lhs, uint64_t rhs)
-        if(roundup)
+        if(roundup || ((ansf & 3) == 3))
@@ -194 +194 @@ uint64_t add(uint64_t lhs, uint64_t rhs)
-    if(ansf >= (1 << 53)){
+    if(ansf >= (1ull << 53)){
@@ -200 +200 @@ uint64_t add(uint64_t lhs, uint64_t rhs)
-    if(ansexp == 0 && ansf >= (1 << 52))
+    if(ansexp == 0 && ansf >= (1ull << 52))
@@ -203 +203 @@ uint64_t add(uint64_t lhs, uint64_t rhs)
-    assert((ansexp != 0 && ansf < (1 << 53)) || (ansexp == 0 && ansf < (1 << 52)));
+    assert((ansexp != 0 && ansf < (1ull << 53)) || (ansexp == 0 && ansf < (1ull << 52)));
@@ -205 +205 @@ uint64_t add(uint64_t lhs, uint64_t rhs)
-    if(ansexp >= ((1 << 11) - 1)) // overflow
+    if(ansexp >= ((1ull << 11) - 1)) // overflow
@@ -208 +208 @@ uint64_t add(uint64_t lhs, uint64_t rhs)
-        ans = ansexp << 52 | (ansf & ((1 << 52) - 1));
+        ans = ansexp << 52 | (ansf & ((1ull << 52) - 1));
@@ -210 +210 @@ uint64_t add(uint64_t lhs, uint64_t rhs)
-    ans |= (1 << 63) & lhs; // Add sign
+    ans |= (1ull << 63) & lhs; // Add sign
@@ -246 +246 @@ uint64_t subtract(uint64_t lhs, uint64_t
-        uint64_t cur = LowBit(rhsf) >> ediff;
+        uint64_t cur = (ediff < 64) ? (LowBit(rhsf) >> ediff) : 0;
@@ -255 +255 @@ uint64_t subtract(uint64_t lhs, uint64_t
-    while(ansexp > 0 && (ansf & (1 << 54)) == 0){
+    while(ansexp > 0 && (ansf & (1ull << 54)) == 0){
@@ -272 +272 @@ uint64_t subtract(uint64_t lhs, uint64_t
-    if(ansf >= (1 << 53)){
+    if(ansf >= (1ull << 53)){
@@ -277 +277 @@ uint64_t subtract(uint64_t lhs, uint64_t
-    if(ansexp == 0 && ansf >= (1 << 52))
+    if(ansexp == 0 && ansf >= (1ull << 52))
@@ -280 +280 @@ uint64_t subtract(uint64_t lhs, uint64_t
-    ans = ansexp << 52 | (ansf & ((1 << 52) - 1));
+    ans = ansexp << 52 | (ansf & ((1ull << 52) - 1));
@@ -282 +282 @@ uint64_t subtract(uint64_t lhs, uint64_t
-    ans |= lhs & (1 << 63); // Add sign
```

```
+    ans |= lhs & (1ull << 63); // Add sign
@@ -297 +297 @@ uint64_t multiply(uint64_t lhs, uint64_t
-    int64_t ansexp = Exp(lhs) + Exp(rhs) - 1023 - 51;
+    int64_t ansexp = Exp(lhs) + Exp(rhs) - 1023ll - 51ll;
@@ -301 +301 @@ uint64_t multiply(uint64_t lhs, uint64_t
-    while(ansexp < 0 || ansf >= (1 << 54)){
+    while(ansexp < 0 || ansf >= (1ull << 54)){
@@ -306 +306 @@ uint64_t multiply(uint64_t lhs, uint64_t
-    while(ansexp > 0 && (ansf & (1 << 53)) == 0){
+    while(ansexp > 0 && (ansf & (1ull << 53)) == 0){
@@ -330 +330 @@ uint64_t multiply(uint64_t lhs, uint64_t
-    if(ansf >= (1 << 53)){
+    if(ansf >= (1ull << 53)){
@@ -336 +336 @@ uint64_t multiply(uint64_t lhs, uint64_t
-    if(ansexp >= ((1 << 11) - 1)) // overflow
+    if(ansexp >= ((1ull << 11) - 1)) // overflow
@@ -339 +339 @@ uint64_t multiply(uint64_t lhs, uint64_t
-        ans = ansexp << 52 | (ansf & ((1 << 52) - 1));
+        ans = ansexp << 52 | (ansf & ((1ull << 52) - 1));
@@ -341 +341,2 @@ uint64_t multiply(uint64_t lhs, uint64_t
-    ans |= ((1 << 63) & lhs) ^ ((1 << 63) & rhs); // Add sign
+    ans |= ((1ull << 63) & lhs) ^ ((1ull << 63) & rhs); // Add sign
+    //printf("%.120lf\n", ans);
@@ -351,0 +353,2 @@ uint64_t divide(uint64_t lhs, uint64_t r
+        else if ((1ull << 63) & (lhs ^ rhs))
+            return NINF;
@@ -359 +362 @@ uint64_t divide(uint64_t lhs, uint64_t r
-            return ((1 << 63) & (lhs ^ rhs)); // signed zero
+            return ((1ull << 63) & (lhs ^ rhs)); // signed zero
@@ -361 +364,4 @@ uint64_t divide(uint64_t lhs, uint64_t r
-    if(isINF(lhs))
+    if(isINF(lhs)){
+        if ((1ull << 63) & (lhs ^ rhs))
+            return NINF;
+        else
@@ -362,0 +369 @@ uint64_t divide(uint64_t lhs, uint64_t r
+    }
@@ -366,2 +373,2 @@ uint64_t divide(uint64_t lhs, uint64_t r
-    int64_t ansexp = Exp(lhs) - Exp(rhs) + 1023;
-    uint64_t ansf = ((intEx)(Fraction(lhs)) << 54) / (intEx)(Fraction(rhs));
+    int64_t ansexp = Exp(lhs) - Exp(rhs) + 1023ll;
+    uint64_t ansf = (((intEx)(Fraction(lhs))) << 54) / (intEx)(Fraction(rhs));
@@ -369 +376 @@ uint64_t divide(uint64_t lhs, uint64_t r
-    if(((intEx)(Fraction(lhs)) << 54) % (intEx)(Fraction(rhs)) != 0)
+    if((((intEx)(Fraction(lhs))) << 54) % (intEx)(Fraction(rhs)) != 0)
@@ -373 +380 @@ uint64_t divide(uint64_t lhs, uint64_t r
-    while(ansexp < 0 || ansf >= (1 << 55)){
+    while(ansexp < 0 || ansf >= (1ull << 55)){
@@ -378 +385 @@ uint64_t divide(uint64_t lhs, uint64_t r
-    while(ansexp > 0 && (ansf & (1 << 54)) == 0){
+    while(ansexp > 0 && (ansf & (1ull << 54)) == 0){
@@ -404 +411 @@ uint64_t divide(uint64_t lhs, uint64_t r
-    if(ansf >= (1 << 53)){
+    if(ansf >= (1ull << 53)){
@@ -410 +417 @@ uint64_t divide(uint64_t lhs, uint64_t r
-    if(ansexp >= ((1 << 11) - 1)) // overflow
+    if(ansexp >= ((1ull << 11) - 1)) // overflow
@@ -413 +420 @@ uint64_t divide(uint64_t lhs, uint64_t r
-        ans = ansexp << 52 | (ansf & ((1 << 52) - 1));
+        ans = ansexp << 52 | (ansf & ((1ull << 52) - 1));
@@ -415 +422,2 @@ uint64_t divide(uint64_t lhs, uint64_t r
-    ans |= ((1 << 63) & lhs) ^ ((1 << 63) & rhs); // Add sign
+    ans |= (1ull << 63) & (lhs ^ rhs); // Add sign
+    //printf("%lf\n", ans);
@@ -459 +467 @@ uint64_t read_from_string(char* str){
-    sscanf(str, "%lf", &x);
+    sscanf(str, "%lf", (double *)&x);
@@ -467,0 +476,3 @@ char* write_to_string(uint64_t x){
+        if(x & (1ull << 63)) {
+            strcpy(ans, "-inf");
+        } else {
@@ -468,0 +480 @@ char* write_to_string(uint64_t x){
+        }
@@ -475 +487 @@ inline uint64_t LowBit(uint64_t x){
-    return x & ((~x) + 1);
+    return x & ((~x) + 1ull);
@@ -479 +491 @@ inline uint64_t Negative(uint64_t x){
```

```
-    return isNaN(x)? x : (x ^ (1 << 63));
+    return isNaN(x)? x : (x ^ (1ull << 63));
@@ -483 +495 @@ inline int64_t Exp(uint64_t x){
-    return (x >> 52) & ((1 << 11) - 1);
+    return (x >> 52) & ((1ull << 11) - 1);
@@ -492 +504 @@ inline uint64_t Fraction(uint64_t x){
-        return 1 << 52 | (x & ((1 << 52) - 1));
+        return 1ull << 52 | (x & ((1ull << 52) - 1));
@@ -494 +506 @@ inline uint64_t Fraction(uint64_t x){
-        return (x & ((1 << 52) - 1)) << 1; // normalize subnormal
+        return (x & ((1ull << 52) - 1)) << 1; // normalize subnormal

```