BugLab 实验报告

A. shuffle

调试思路

此题共两个bug， 首先代码在判断边界时混淆了n与m，其次位运算对两数的交换

无法处理交换位置相同的情况，需要特判交换位置。

代码补丁

```
shuffle.cpp
--- buggy/shuffle.cpp   2022-10-07 03:44:31.000000000 -0400
+++ fixed/shuffle.cpp   2022-10-07 08:44:21.726542908 -0400
@@ -21,10 +21,11 @@ int main() {
     for(int i = 0; i < n; i++) {
         int a, b;
         std::cin >> a >> b;
-        if(a < 0 || a >= n || b < 0 || b >= n) {
+        if(a < 0 || a >= m || b < 0 || b >= m) {
             std::cout << -1 << std::endl;
             return 0;
         }
+        if(a == b) continue;
         swap(data[a], data[b]);
     }
```

B. polycalc

调试思路

本题共两个bug，首先作为返回值，在函数calcOne(Node)中result没有赋初值。其次，node.exp类型是unsigned int，在循环中当node.exp为0时，node.exp - 1为tmax，会导致程序错误。

代码补丁

```
polycalc.cpp
--- buggy/polycalc.cpp  2022-10-07 03:44:31.000000000 -0400
+++ fixed/polycalc.cpp  2022-10-07 08:54:24.671701584 -0400
@@ -32,7 +32,7 @@ ElemTypeB calc() {
         data[i] = Node(a, b);
     }

-    ElemTypeB result;
+    ElemTypeB result = 0;
     for(int i = 0; i < n; i++) {
         result += calcOnce(data[i]);
     }
```

```
@@ -42,8 +42,8 @@ ElemTypeB calc() {
 }

 ElemTypeB calcOnce(Node node) {
-    ElemTypeB result;
-    for(int i = 0; i <= node.exp - 1; i++) {
+    ElemTypeB result = 1;
+    for(int i = 0; i < node.exp; i++) {
         result *= node.base;
     }
```

C. violetStore

调试思路

本题bug大都为指针错误。首先，malloc函数只会为指针分配空间而不会调用类的构造函数，其次，在构造函数中没有为n赋初值，第三个bug为在为指针所指的空间赋值时，将指针移动到了数组的尾部，导致非法内存的访问，最后销毁指针数组时需要用delete[]析构指针数组中的内容。

代码补丁

```
violetStore.cpp
--- buggy/violetStore.cpp   2022-10-07 03:44:31.000000000 -0400
+++ fixed/violetStore.cpp   2022-10-18 07:08:27.776630033 -0400
@@ -2,31 +2,35 @@
 #include <string>
 #define min(a,b) a<=b?a:b

-class price
+class price
 {
 private:
     int n;                  // item's number
     std::string* items;     // item's name
     int* prices;            // item's price
     std::string store;      // store's name
+    int cnt;
 public:
     price()
     {
-        store = "Violet";
+        n = 3;
+        store = "Violet";
         items = new std::string[3]; // I'm poor, max 3 items
         prices = new int[3];
+        cnt = 0;
     }
     ~price()
     {
```

```
-        delete items;
-        delete prices;
+        delete[] items;
+        delete[] prices;
     }
-    void add_item(std::string name, int price)
+    void add_item(std::string name, int price)
     {
-        *items++ = name;
-        *prices++ = price;
+        items[cnt] = name;
+        prices[cnt] = price;
+        cnt ++ ;
     }
-    void print_items()
+    void print_items()
     {
         std::cout << "items of " << store << ":\n";
         for(int i = 0; i < n; ++i)
@@ -40,7 +44,7 @@ public:

 int main()
 {
-    price* test = (price*)malloc(sizeof(price));
+    price* test = new price();
     test->add_item("apple", 1);
     test->add_item("banana", 3);
     test->add_item("cherry", 2);
```

D. swapCase

调试思路

本题共有两个bug。首先，scanf无法读取空格，当读取到空格时，会将字符串截断，需要用fgets来读取整行字符串。其次，strlen函数求字符串长度的时间复杂度为O(n), 在循环判断时使用会使整体时间复杂度退化为O(n^2), 需要将字符串长度的函数放在循环外执行。

代码补丁

```
swapCase.cpp
--- buggy/swapCase.cpp  2022-10-07 03:44:31.000000000 -0400
+++ fixed/swapCase.cpp  2022-10-10 21:22:05.650736393 -0400
@@ -5,8 +5,8 @@ const int MAXN=1e5+10;
 char s[MAXN];

 int main(){
-    scanf("%s", s);
-    for(int i = 0; i < strlen(s); ++i){
+    fgets(s, MAXN, stdin);
```

```
+    for(int i = 0; s[i]; ++i){
         if(std::isalpha(s[i]))
             s[i] ^= 1<<5;
     }
```

E. xorsum

调试思路

本题共一个bug，在同一行调用ReadInt函数时，先调用右边再调用左边，导致调用Replace函数时pos与value的位置交换，需要在函数内部调用一次swap函数。

代码补丁

```
xorsum.cpp
--- buggy/xorsum.cpp   2022-10-07 03:44:31.000000000 -0400
+++ fixed/xorsum.cpp   2022-10-18 08:46:06.495892755 -0400
@@ -18,6 +18,7 @@ int ReadInt(){
 }

 void Replace(int pos, int value){
+    std::swap(pos, value);
     ans ^= v[pos] ^ value;
     v[pos] = value;
 }
```

F. mergeIntervals

调试思路

本题共两个bug。首先在对区间进行排序的时候应当先以左边界为第一关键字，以右边界为第二关键字排序。其次当下一个区间的左边界落在上个合并后的区间右边界的左边时，合并后区间的右边界应该为该区间的右边界与上个右边界的最大值。

代码补丁

```
mergeIntervals.cpp
--- buggy/mergeIntervals.cpp   2022-10-07 03:44:31.000000000 -0400
+++ fixed/mergeIntervals.cpp   2022-10-09 06:16:38.485366582 -0400
@@ -6,7 +6,8 @@ struct Range{
 };

 bool compare(const Range& x, const Range& y){
-    return x.l <= y.l;
+    if(x.l != y.l) return x.l < y.l;
+    return x.r < y.r;
 }

 int main(){
@@ -23,7 +24,7 @@ int main(){
     Range last = R[0];
     for(auto it = std::next(R.begin()); it != R.end(); ++it){
         if(it->l <= last.r)
-            last.r = it->r;
+            last.r = std::max(last.r, it->r);
         else{
             ans.push_back(last);
             last = *it;
```

G. 8num

调试思路

本题共两个bug。首先stack中的状态需要通过new的方式向系统动态申请内存，否则当curstate从stack中弹出时，之前创建的state对象占用的内存块会被系统回收，为下一个state对象提供空间，导致state中指向其父节点的指针指向的内存块中的内容不再是原本其父节点的内容。其次，返回路径时，不能将转移后的指针free，同样会使free已经转移的指针，导致空指针错误。

代码补丁

```
8num.cpp
--- buggy/8num.cpp      2022-10-07 03:44:31.000000000 -0400
+++ fixed/8num.cpp      2022-10-08 06:25:10.522853696 -0400
@@ -84,8 +84,7 @@ int IDS(int max_depth){
         //     2）移动空格位置  nextState->zero_x = curState->zero_x+move_x[i]; nextState-
>zero_y = curState->zero_y+move_y[i];
         //     3）判断空格移动是否在允许范围 (nextState->zero_x<0||nextState-
>zero_x>2||nextState->zero_y<0||nextState->zero_y>2)
         //     4）防止状态重复
-        State curs = State(que.top().first);
-        State* curState = &curs;
+        State* curState = new State(que.top().first);
```

```
        curDepth = que.top().second;
        que.pop();
        node_num--;
@@ -95,7 +94,6 @@ int IDS(int max_depth){
            while(curState){
                stack_path.push(*curState);
                curState = curState->parent;
-               free(curState);
            }
            return steps;
        }
```

H. segtree

调试思路

本题共有两个bug。首先结构体内的变量未全部初始化，需将指针指向NULL并将sum和add的值都初始化为0。同时，在递归取每一个节点的左右儿子时，需要判断指向其左右儿子的指针是否为NULL。其次，因为本题的数据范围为1e5，每次修改的的数k的取值范围为1e9，会导致int范围无法正确表示，需要将区间和的返回值和结构体中的sum与add的数据类型改为long long int 类型。

代码补丁

```
segtree.cpp
--- buggy/segtree.cpp  2022-10-07 03:44:31.000000000 -0400
+++ fixed/segtree.cpp  2022-10-08 08:02:52.922797524 -0400
@@ -6,59 +6,61 @@

 const int MAXN=1e5+10;

+typedef long long LL;
+
 struct Node{
-  int l;
-  int r;
-  int add;
-  int sum;
-  int size;
-  Node* lch;
-  Node* rch;
+    int l;
+    int r;
+    LL add = 0;
+    LL sum = 0;
+    int size;
+    Node* lch = nullptr;
+    Node* rch = nullptr;
     ~Node();
-  Node(int,int);
-  void Maintain();
-  void PushDown();
```

```
-   int Query(const int&,const int&);
-   void Add(const int&,const int&,const int&);
+     Node(int,int);
+     void Maintain();
+     void PushDown();
+     LL Query(const int&,const int&);
+     void Add(const int&,const int&,const int&);
 };

 int data[MAXN];

 int main(){
-   int n,q;
-   scanf("%d%d",&n,&q);
-   for(int i=1;i<=n;i++)
-       scanf("%d",data+i);
-   Node* N=new Node(1,n);
-   int t,l,r,d;
-   for(int i=0;i<q;i++){
-       scanf("%d",&t);
-       if(t==1){
-           scanf("%d%d%d",&l,&r,&d);
-           N->Add(l,r,d);
-       }
-       else if(t==2){
-           scanf("%d%d",&l,&r);
-           printf("%d\n",N->Query(l,r));
-       }
-   }
+     int n,q;
+     scanf("%d%d",&n,&q);
+     for(int i=1;i<=n;i++)
+         scanf("%d",data+i);
+     Node* N=new Node(1,n);
+     int t, l,r,d;
+     for(int i=0;i<q;i++){
+         scanf("%d", &t);
+         if(t==1){
+             scanf("%d%d%d",&l,&r,&d);
+             N->Add(l,r,d);
+         }
+         else{
+             scanf("%d%d",&l,&r);
+             printf("%lld\n",N->Query(l,r));
+         }
+     }
      delete N;
-   return 0;
+     return 0;
 }

 Node::Node(int l,int r){
-   this->l=l;
-   this->r=r;
-   this->add=0;
-   this->size=r-l+1;
-   if(l==r)
-       this->sum=data[r];
```

```
-  else{
-     int mid=(l+r)>>1;
-     this->lch=new Node(l,mid);
-     this->rch=new Node(mid+1,r);
-     this->Maintain();
-  }
+    this->l=l;
+    this->r=r;
+    this->add=0;
+    this->size=r-l+1;
+    if(l==r)
+        this->sum=data[r];
+    else{
+        int mid=(l+r)>>1;
+        this->lch=new Node(l,mid);
+        this->rch=new Node(mid+1,r);
+        this->Maintain();
+    }
 }

 Node::~Node(){
@@ -69,43 +71,48 @@ Node::~Node(){
 }

 void Node::Add(const int& l,const int& r,const int& d){
-  if(l<=this->l&&this->r<=r){
-     this->add+=d;
-     this->sum+=d*this->size;
-  }
-  else{
-     this->PushDown();
-     if(l<=this->lch->r)
-        this->lch->Add(l,r,d);
-     if(this->rch->l<=r)
-        this->rch->Add(l,r,d);
-     this->Maintain();
-  }
-}
-
-int Node::Query(const int& l,const int& r){
-  if(l<=this->l&&this->r<=r)
-     return this->sum;
-  else{
-     this->PushDown();
-     if(r<=this->lch->r)
-        return this->lch->Query(l,r);
-     if(this->rch->l<=l)
-        return this->rch->Query(l,r);
-     return this->lch->Query(l,r)+this->rch->Query(l,r);
-  }
+    if(l<=this->l&&this->r<=r){
+        this->add+=d;
+        this->sum+=d*this->size;
+    }
+    else{
+        this->PushDown();
+        if(this -> lch != nullptr && l<=this->lch->r)
+            this->lch->Add(l,r,d);
```

```
+        if(this -> rch != nullptr && this->rch->l<=r)
+            this->rch->Add(l,r,d);
+        this->Maintain();
+    }
+}
+
+LL Node::Query(const int& l,const int& r){
+    if(l<=this->l&&this->r<=r)
+        return this->sum;
+    else{
+        this->PushDown();
+        if(this -> lch != nullptr && r<=this->lch->r)
+            return this->lch->Query(l,r);
+        if(this -> rch != nullptr && this->rch->l<=l)
+            return this->rch->Query(l,r);
+        LL res = 0;
+        if(this -> lch != nullptr) res += this->lch->Query(l,r);
+        if(this -> rch != nullptr) res += this->rch->Query(l,r);
+        return res;
+    }
 }

 inline void Node::Maintain(){
-  this->sum=this->lch->sum+this->rch->sum;
+    this -> sum = 0;
+    if(this -> lch != nullptr) this -> sum += this->lch->sum;
+    if(this -> rch != nullptr) this -> sum += this->rch->sum;
 }

 inline void Node::PushDown(){
-  if(this->add){
-     this->lch->add+=this->add;
-     this->rch->add+=this->add;
-     this->lch->sum+=this->add*this->lch->size;
-     this->rch->sum+=this->add*this->rch->size;
-     this->add=0;
-  }
+    if(this->add){
+        if(this -> lch != nullptr) this->lch->add+=this->add;
+        if(this -> rch != nullptr) this->rch->add+=this->add;
+        if(this -> lch != nullptr) this->lch->sum+=this->add*this->lch->size;
+        if(this -> rch != nullptr) this->rch->sum+=this->add*this->rch->size;
+        this->add=0;
+    }
 }
```

I. antbuster

调试思路

本题共三处bug。首先在计算斜率的时候没有考虑斜率不存在的情况，需要加上特判;其次在蚂蚁特殊移动SpecialMove的函数中，求逆时针旋转时，为了避免出现负数，每次计算需要加上4;最后每秒钟信号素衰减应在有信号素的格子中进行，而不是每一个格子。

代码补丁

```
antbuster.cpp
--- buggy/antbuster.cpp 2022-10-07 03:44:31.000000000 -0400
+++ fixed/antbuster.cpp 2022-10-18 09:28:51.670600229 -0400
@@ -112,10 +112,15 @@ void Tower::Fire(){
         return;
     double dx=this->x-target->x;
     double dy=this->y-target->y;
-    double k=dy/dx;
-    double b=this->y-k*this->x;
+    double k, b, p = -1.0;
+    if(fabs(dx) > 0.1)
+    {
+        k=dy/dx;
+        b=this->y-k*this->x;
+    }
+    else k = 1, b = -this -> x, p = 0;
     for(std::list<Ant>::iterator i=ants.begin();i!=ants.end();++i){
-        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(k,p,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)){
             i->HP-=d;
         }
     }
@@ -162,9 +167,9 @@ void Ant::NormalMove(int dir){
 }

 void Ant::SpecialMove(int dir){
-    dir=(dir-1)%4;
+    dir=(dir+3)%4;
     while(!this->CheckAvailable(this->x+dx[dir],this->y+dy[dir]))
-        dir=(dir-1)%4;
+        dir=(dir+3)%4;
     this->NormalMove(dir);
 }

@@ -196,7 +201,8 @@ void CleanDeath(){
 void DecreaseSignal(){
     for(int i=0;i<=n;i++){
         for(int j=0;j<=m;j++){
-            --sign[i][j];
+            if(sign[i][j] > 0)
+                --sign[i][j];
         }
     }
 }
```

J. softDouble

调试思路

本题共五个bug。第一，为*mm_cvtsi64_si128函数的问题，此函数的作用是将传入的参数中64位设置在128位中的高64位，需要将其改为*mm_set_pi64x将二进制码的解析方式转化为double；第二，到1左移的位数大于31位时，会超出int所能表示的数据范围，需要将1的类型强制转化为long long int。第三，在算加减法时，所得到的两数exp之差可能大于63，产生非法移位，需要将ediff的最大值设为63。第四，判断inf时没有考虑符号问题，在最后将结果转化为字符串时，需要通过符号位将+inf与-inf区分。第五，在加法计算roundup时，没有考虑向偶数舍入的情况，需要补充计算。

代码补丁

```
softDouble.cpp
--- buggy/softDouble.cpp    2022-10-07 03:44:31.000000000 -0400
+++ fixed/softDouble.cpp    2022-10-14 01:58:50.064333552 -0400
@@ -122,15 +122,15 @@ int main(){ // Function: Parse & Evaluat
 }

 inline bool isNaN(uint64_t x){
-    return (Exp(x) == (1 << 11) - 1) && (Fraction(x) & ((1 << 52) - 1)) != 0;
+    return (Exp(x) == (1 << 11) - 1) && (Fraction(x) & ((1ll << 52) - 1)) != 0;
 }

 inline bool isINF(uint64_t x){
-    return (Exp(x) == (1 << 11) - 1) && (Fraction(x) & ((1 << 52) - 1)) == 0;
+    return (Exp(x) == (1 << 11) - 1) && (Fraction(x) & ((1ll << 52) - 1)) == 0;
 }

 inline bool isZero(uint64_t x){
-    return (x & ((1 << 63) - 1)) == 0;
+    return (x & ((1ll << 63) - 1)) == 0;
 }

 uint64_t add(uint64_t lhs, uint64_t rhs){
@@ -153,7 +153,7 @@ uint64_t add(uint64_t lhs, uint64_t rhs)
         std::swap(lhs, rhs);
     }

-    int ediff = Exp(lhs) - Exp(rhs);
+    int ediff = std::min(63ll, Exp(lhs) - Exp(rhs));
     assert(ediff >= 0);

     uint64_t ans = 0;
@@ -172,7 +172,7 @@ uint64_t add(uint64_t lhs, uint64_t rhs)
     }

     // Adjust EXP
-    while(ansf >= (1 << 54)){
+    while(ansf >= (1ll << 54)){
```

```
         roundup = roundup || (ansf & 1) != 0;
         ansf >>= 1;
         ++ansexp;
@@ -180,34 +180,34 @@ uint64_t add(uint64_t lhs, uint64_t rhs)
     if(ansexp == 0){ // subnormalized
         assert((ansf & 1) == 0);
         ansf >>= 1;
-        assert(ansexp < (1 << 53));
+        assert(ansexp < (1ll << 53));
     }
     // Rounding
     if((ansf & 1) == 0)
         ansf >>= 1;
     else{
         ansf >>= 1;
-        if(roundup)
+        if(roundup || ansf & 1)
             ++ansf;
     }
     // NOTE: only 011111 -> 100000, no more rounding required
-    if(ansf >= (1 << 53)){
+    if(ansf >= (1ll << 53)){
         assert(ansexp != 0);
         assert((ansf & 1) == 0);
         ansf >>= 1;
         ++ansexp;
     }
-    if(ansexp == 0 && ansf >= (1 << 52))
+    if(ansexp == 0 && ansf >= (1ll << 52))
         ++ansexp;

-    assert((ansexp != 0 && ansf < (1 << 53)) || (ansexp == 0 && ansf < (1 << 52)));
+    assert((ansexp != 0 && ansf < (1ll << 53)) || (ansexp == 0 && ansf < (1ll << 52)));

     if(ansexp >= ((1 << 11) - 1)) // overflow
         ans = INF;
     else
-        ans = ansexp << 52 | (ansf & ((1 << 52) - 1));
+        ans = ansexp << 52 | (ansf & ((1ll << 52) - 1));

-    ans |= (1 << 63) & lhs; // Add sign
+    ans |= (1ll << 63) & lhs; // Add sign
     return ans;
 }

@@ -233,7 +233,7 @@ uint64_t subtract(uint64_t lhs, uint64_t
         std::swap(lhs, rhs);
     }

-    int ediff = Exp(lhs) - Exp(rhs);
+    int ediff = std::min(Exp(lhs) - Exp(rhs), 63ll);
     assert(ediff >= 0);

     uint64_t ans = 0;
@@ -252,7 +252,7 @@ uint64_t subtract(uint64_t lhs, uint64_t
     }

     // Adjust EXP
```

```
-    while(ansexp > 0 && (ansf & (1 << 54)) == 0){
+    while(ansexp > 0 && (ansf & (1ll << 54)) == 0){
         --ansexp;
         ansf <<= 1;
     }
@@ -269,17 +269,17 @@ uint64_t subtract(uint64_t lhs, uint64_t
             ++ansf;
     }
     // NOTE: only 011111 -> 100000, no more rounding required
-    if(ansf >= (1 << 53)){
+    if(ansf >= (1ll << 53)){
         if(ansexp > 0)
             ansf >>= 1;
         ++ansexp;
     }
-    if(ansexp == 0 && ansf >= (1 << 52))
+    if(ansexp == 0 && ansf >= (1ll << 52))
         ++ansexp;

-    ans = ansexp << 52 | (ansf & ((1 << 52) - 1));
+    ans = ansexp << 52 | (ansf & ((1ll << 52) - 1));

-    ans |= lhs & (1 << 63); // Add sign
+    ans |= lhs & (1ll << 63); // Add sign
     return negflag ? Negative(ans) : ans;
 }

@@ -298,12 +298,12 @@ uint64_t multiply(uint64_t lhs, uint64_t
     intEx ansf = ((intEx)(Fraction(lhs)) * (intEx)(Fraction(rhs)));

     // Adjusting exp
-    while(ansexp < 0 || ansf >= (1 << 54)){
+    while(ansexp < 0 || ansf >= (1ll << 54)){
         ++ansexp;
         roundup |= ansf & 1;
         ansf >>= 1;
     }
-    while(ansexp > 0 && (ansf & (1 << 53)) == 0){
+    while(ansexp > 0 && (ansf & (1ll << 53)) == 0){
         --ansexp;
         ansf <<= 1;
     }
@@ -314,7 +314,6 @@ uint64_t multiply(uint64_t lhs, uint64_t
     }

     if(ansexp == 0){ // subnormal handling
-        roundup |= ansf & 1;
         ansf >>= 1;
     }

@@ -327,18 +326,19 @@ uint64_t multiply(uint64_t lhs, uint64_t
             ++ansf;
     }

-    if(ansf >= (1 << 53)){
+    if(ansf >= (1ll << 53)){
         if(ansexp > 0)
             ansf >>= 1;
```

```
         ++ansexp;
     }

+
     if(ansexp >= ((1 << 11) - 1)) // overflow
         ans = INF;
     else
-        ans = ansexp << 52 | (ansf & ((1 << 52) - 1));
+        ans = ansexp << 52 | (ansf & ((1ll << 52) - 1));

-    ans |= ((1 << 63) & lhs) ^ ((1 << 63) & rhs); // Add sign
+    ans |= ((1ll << 63) & lhs) ^ ((1ll << 63) & rhs); // Add sign
     return ans;
 }

@@ -350,32 +350,32 @@ uint64_t divide(uint64_t lhs, uint64_t r
         if(isZero(lhs))
             return NaN;
         else
-            return INF;
+            return INF | ((lhs ^ rhs) & (1ll << 63));
     }
     if(isINF(rhs)){  // divided by INF
         if(isINF(lhs))
             return NaN;
         else
-            return ((1 << 63) & (lhs ^ rhs)); // signed zero
+            return ((1ll << 63) & (lhs ^ rhs)); // signed zero
     }
     if(isINF(lhs))
-        return INF; // INF/INF handled, other return INF
+        return INF | ((lhs ^ rhs) & (1ll << 63)); // INF/INF handled, other return INF

     uint64_t ans = 0;
     bool roundup = false;
     int64_t ansexp = Exp(lhs) - Exp(rhs) + 1023;
-    uint64_t ansf = ((intEx)(Fraction(lhs)) << 54) / (intEx)(Fraction(rhs));
+    intEx ansf = ((intEx)(Fraction(lhs)) << 54) / (intEx)(Fraction(rhs));

     if(((intEx)(Fraction(lhs)) << 54) % (intEx)(Fraction(rhs)) != 0)
         roundup = true;

     // Adjusting exp
-    while(ansexp < 0 || ansf >= (1 << 55)){
+    while(ansexp < 0 || ansf >= (1ll << 55)){
         ++ansexp;
         roundup |= ansf & 1;
         ansf >>= 1;
     }
-    while(ansexp > 0 && (ansf & (1 << 54)) == 0){
+    while(ansexp > 0 && (ansf & (1ll << 54)) == 0){
         --ansexp;
         ansf <<= 1;
     }
@@ -401,7 +401,7 @@ uint64_t divide(uint64_t lhs, uint64_t r
             ++ansf;
     }
```

```
-    if(ansf >= (1 << 53)){
+    if(ansf >= (1ll << 53)){
         if(ansexp > 0)
             ansf >>= 1;
         ++ansexp;
@@ -410,9 +410,9 @@ uint64_t divide(uint64_t lhs, uint64_t r
     if(ansexp >= ((1 << 11) - 1)) // overflow
         ans = INF;
     else
-        ans = ansexp << 52 | (ansf & ((1 << 52) - 1));
+        ans = ansexp << 52 | (ansf & ((1ll << 52) - 1));

-    ans |= ((1 << 63) & lhs) ^ ((1 << 63) & rhs); // Add sign
+    ans |= ((1ll << 63) & lhs) ^ ((1ll << 63) & rhs); // Add sign
     return ans;
 }

@@ -465,9 +465,10 @@ char* write_to_string(uint64_t x){
     if(isNaN(x))
         strcpy(ans, "nan");
     else if(isINF(x))
-        strcpy(ans, "inf");
+        if(x >> 63 & 1) strcpy(ans, "-inf");
+        else strcpy(ans, "inf");
     else
-        sprintf(ans, "%.1200f", _mm_cvtsi64_si128(x));
+        sprintf(ans, "%.1200lf",_mm_set_pi64x(x));
     return ans;
 }

@@ -476,11 +477,11 @@ inline uint64_t LowBit(uint64_t x){
 }

 inline uint64_t Negative(uint64_t x){
-    return isNaN(x)? x : (x ^ (1 << 63));
+    return isNaN(x)? x : (x ^ (1ll << 63));
 }

 inline int64_t Exp(uint64_t x){
-    return (x >> 52) & ((1 << 11) - 1);
+    return (x >> 52) & ((1ll << 11) - 1);
 }

 inline int Sign(uint64_t x){
@@ -489,7 +490,7 @@ inline int Sign(uint64_t x){

 inline uint64_t Fraction(uint64_t x){
     if(Exp(x) != 0)
-        return 1 << 52 | (x & ((1 << 52) - 1));
+        return 1ll << 52 | (x & ((1ll << 52) - 1));
     else
-        return (x & ((1 << 52) - 1)) << 1; // normalize subnormal
+        return (x & ((1ll << 52) - 1)) << 1; // normalize subnormal
 }
```