string Multiply(string a, string b) {

int kA = 0, indexA = -1;

int kB = 0, indexB = -1;

CountCommas(kA, indexA, a);

CountCommas(kB, indexB, b);

// Remove decimal points and track their positions

int decimalPlacesA = (kA == 1) ? (a.size() - 1 - indexA) : 0;

int decimalPlacesB = (kB == 1) ? (b.size() - 1 - indexB) : 0;

if (kA == 1) { ShiftForDBT(a, indexA); a = Del(a); }

if (kB == 1) { ShiftForDBT(b, indexB); b = Del(b); }

int len1 = a.size(), len2 = b.size();

string result(len1 + len2, '0');

for (int i = len1 - 1; i >= 0; i--) {

int carry = 0;

for (int j = len2 - 1; j >= 0; j--) {

int mul = (a[i] - '0') \* (b[j] - '0') + (result[i + j + 1] - '0') + carry;

carry = mul / 10;

result[i + j + 1] = (mul % 10) + '0';

}

result[i] += carry;

}

// Adjust for decimal places

int totalDecimals = decimalPlacesA + decimalPlacesB;

if (totalDecimals > 0) {

ReverseString(result);

while (result.size() <= totalDecimals) result += '0';

result = '.' + result;

for (int i = 0; i < totalDecimals; i++) {

char temp = result[i];

result[i] = result[i + 1];

result[i + 1] = temp;

}

ReverseString(result);

result = Del(result); // Remove trailing dot if present

}

// Remove leading zeros

if (result[0] == '0') {

int i = 0, k = 0;

return RemoveZeros(result, i, k);

}

return result;

}

string DivByTwo(string num) {

int k = 0, index = -1;

CountCommas(k, index, num);

int decimalPlaces = (k == 1) ? (num.size() - 1 - index) : 0;

if (k == 1) { ShiftForDBT(num, index); num = Del(num); }

string result;

int carry = 0;

for (int i = 0; i < num.size(); i++) {

int curr = carry \* 10 + (num[i] - '0');

result += (curr / 2) + '0';

carry = curr % 2;

}

if (carry) result += '5'; // If there's a remainder, append 0.5 as "5"

// Adjust decimal point

if (decimalPlaces > 0 || carry) {

ReverseString(result);

while (result.size() <= decimalPlaces + (carry ? 1 : 0)) result += '0';

result = '.' + result;

for (int i = 0; i < decimalPlaces + (carry ? 1 : 0); i++) {

char temp = result[i];

result[i] = result[i + 1];

result[i + 1] = temp;

}

ReverseString(result);

result = Del(result);

}

// Remove leading/trailing zeros

if (result[0] == '0') {

int i = 0, k = 0;

return RemoveZeros(result, i, k);

}

return result;

}

string Sum(string a, string b) {

int kA = 0, indexA = -1;

int kB = 0, indexB = -1;

CountCommas(kA, indexA, a);

CountCommas(kB, indexB, b);

int diffA = (kA == 1) ? (a.size() - 1 - indexA) : 0;

int diffB = (kB == 1) ? (b.size() - 1 - indexB) : 0;

if (kA == 1) { ShiftForDBT(a, indexA); a = Del(a); }

if (kB == 1) { ShiftForDBT(b, indexB); b = Del(b); }

// Align lengths by padding with zeros

while (a.size() < b.size()) a = '0' + a;

while (b.size() < a.size()) b = '0' + b;

int maxDecimals = max(diffA, diffB);

while (a.size() < a.size() + maxDecimals - diffA) a += '0';

while (b.size() < b.size() + maxDecimals - diffB) b += '0';

string result;

int carry = 0;

for (int i = a.size() - 1; i >= 0; i--) {

int sum = (a[i] - '0') + (b[i] - '0') + carry;

result = char(sum % 10 + '0') + result;

carry = sum / 10;

}

if (carry) result = char(carry + '0') + result;

// Reinsert decimal point

if (maxDecimals > 0) {

ReverseString(result);

while (result.size() <= maxDecimals) result += '0';

result = '.' + result;

for (int i = 0; i < maxDecimals; i++) {

char temp = result[i];

result[i] = result[i + 1];

result[i + 1] = temp;

}

ReverseString(result);

result = Del(result);

}

// Remove leading zeros

if (result[0] == '0' && result.size() > 1) {

int i = 0, k = 0;

return RemoveZeros(result, i, k);

}

return result;

}

string SubtractFloat(string a, string b) {

if (!LessThanOrEqual(b, a)) return "0"; // If b > a, return 0

int kA = 0, indexA = -1;

int kB = 0, indexB = -1;

CountCommas(kA, indexA, a);

CountCommas(kB, indexB, b);

int diffA = (kA == 1) ? (a.size() - 1 - indexA) : 0;

int diffB = (kB == 1) ? (b.size() - 1 - indexB) : 0;

if (kA == 1) { ShiftForDBT(a, indexA); a = Del(a); }

if (kB == 1) { ShiftForDBT(b, indexB); b = Del(b); }

// Align lengths

while (a.size() < b.size()) a = '0' + a;

while (b.size() < a.size()) b = '0' + b;

int maxDecimals = max(diffA, diffB);

while (a.size() < a.size() + maxDecimals - diffA) a += '0';

while (b.size() < b.size() + maxDecimals - diffB) b += '0';

string result(a.size(), '0');

int borrow = 0;

for (int i = a.size() - 1; i >= 0; i--) {

int diff = (a[i] - '0') - (b[i] - '0') - borrow;

if (diff < 0) {

diff += 10;

borrow = 1;

} else {

borrow = 0;

}

result[i] = diff + '0';

}

// Reinsert decimal point

if (maxDecimals > 0) {

ReverseString(result);

while (result.size() <= maxDecimals) result += '0';

result = '.' + result;

for (int i = 0; i < maxDecimals; i++) {

char temp = result[i];

result[i] = result[i + 1];

result[i + 1] = temp;

}

ReverseString(result);

result = Del(result);

}

// Remove leading zeros

if (result[0] == '0' && result.size() > 1) {

int i = 0, k = 0;

return RemoveZeros(result, i, k);

}

return result;

}

string Sqrt(string num, int precision) {

if (num == "0" || num == "1") return num;

if (LessThanOrEqual(num, "0")) return "0"; // Handle negative or zero

string result = "0";

string low = "0";

string high = num;

// Binary search for integer part

while (LessThanOrEqual(low, high)) {

string mid = DivByTwo(Sum(low, high));

string square = Multiply(mid, mid);

if (square == num) {

result = mid;

break;

} else if (LessThanOrEqual(square, num)) {

result = mid;

low = Sum(mid, "1");

} else {

high = SubtractFloat(mid, "1");

}

}

// Add decimal precision

string fraction = "0.";

for (int i = 0; i < precision; i++) fraction += "0";

string step = fraction;

for (int i = 0; i < precision; i++) {

string tempResult = Sum(result, step);

string square = Multiply(tempResult, tempResult);

if (LessThanOrEqual(square, num)) {

result = tempResult;

}

step = DivByTwo(step); // Reduce step size

}

return result;

}