#100DaysofCode

Solution

// Instead of multiplying with all digits and adding all at once (which causes memory consumption), we can add every row sequentially by (multiplying x10, x100 ... so on at each level);

Eg:

```
         123
       x  248      24
       -------
level 1     984
level 2    4920   -> x10
level 3   24600   -> x100
       -------
          30504
```

result[i] = carry + (8x3)%10

carry = (8x3)/10

carry = (carry + (8x3)) / 10

```
 123
 x 8   24
 ---
 984
```

Input: 2 vector/array

output: 1 vector.

[reverse the array, to index easily]

```
#include <iostream>
#include <vector>
int main() {
    vector<int> a;
    vector<int> b;
    ~~vector<int> result~~
    int n,m; int t;
    cin>>n>>m;
    for(int i=0; i<n; i++) {
        cin>>t;
        a.push_back(t); }
    for(int i=0; i<m; i++) {
        cin>>t;
        b.push_back(t); }
    vector<int> result(n+m, 0);
    int sign = a.front() < 0 ^ b.front() < 0 ? -1 : 1
```

```
for (int i= n-1 ; i>=0; i--) {
  for (int  j= m-1 ; j>=0; j--) {
    ~~result [i+j+1] += (a[i]* b[j]) %10;~~
    ~~result [i+j] = (a[i]* b[j]) /10;~~
    result [i+j+1] +=  a[i]* b[j] ;
    result [i+j] += result [i+j+1] / 10;
    result [i+j+1] %= 10 ;
  } }
```

counter.

// Remove . Extra '0' characters at the beginning of array.

```
result = vector<int> { find_if_not (begin(result), end(result),
                      [](int a) { return !a; }),
                      end(result) };

if ( result. empty() ) {
    return {0};
}
result. front() *= sign;
return result ;
}
```