Soruda yararlandigimiz gozlem 1'den buyuk en fazla logN adet sayiyi carptigimizda bu carpim degeri N'den kucuk esit olabilir. Bu sayilardan herbiri en az 2 olacagi icin 2^logN'den N buluruz. Yani araliklarin uzunluklari en fazla log(1e18) ~= 60 olabilir. Ayrica 1 olan indisler de bulundugu icin araliklari sayarken bunlari atlamaliyiz. Yani sol bound'u i olarak sabitlersek, j'yi birer birer arttirmak yerine bir sonraki 1 olmayan yeri bularak ilerleyebiliriz. Bunun disinda dizide 0 olmasina gore bir yerden sonra aralikta olmaya baslayabilir. Negatifler icin de prefix carpim sign'i tutmaliyiz.

```
#include <bits/stdc++.h>

using namespace std;

typedef long long ll;
typedef pair < int, int > ii;

const int N = 1e5 + 5;

int n;
ll l, r;
ll a[N];

bool in_range(ll x) {
  bool okay = 1;
  if(r != 42)
    okay &= x <= r;
  if(l != 42)
    okay &= l <= x;
  return okay;
}

ll brute() {
  ll ans = 0;
  for(int i = 1; i <= n; i++) {
    ll mul = 1;
    bool overflow = 0, zero = 0;
    int sign = 1;
    for(int j = i; j <= n; j++) {
      if(abs((double) mul * a[j]) > 5e18)
        overflow = 1;
      mul *= a[j];
      if(a[j] == 0)
        zero = 1;
      if(a[j] < 0)
        sign *= -1;
      if(zero or !overflow) {
        if(in_range(zero ? 0 : mul))
```

```
        ans++;
      }
      else {
        bool okay = 1;
        if(r != 42 and sign == 1)
          okay = 0;
        if(l != 42 and sign == -1)
          okay = 0;
        ans += okay;
      }
    }
  }
  return ans;
}

int next_zero[N], next_bigger[N];
int pre_sign[N], ones[N], zeros[N];

int pos_in(int x, int l, int r) {
  if(pre_sign[x - 1])//should be 1 to be positive
    return ones[r] - ones[l - 1];
  return zeros[r] - zeros[l - 1];
}

int neg_in(int x, int l, int r) {
  if(pre_sign[x - 1])//should be 0 to be negative
    return zeros[r] - zeros[l - 1];
  return ones[r] - ones[l - 1];
}

ll calc() {
  next_zero[n + 1] = n + 1;
  next_bigger[n + 1] = n + 1;
  for(int i = n; i >= 1; i--) {
    next_zero[i] = next_zero[i + 1];
    if(a[i] == 0)
      next_zero[i] = i;
    next_bigger[i] = next_bigger[i + 1];
    if((abs(a[i]) > 1))
      next_bigger[i] = i;
  }
  for(int i = 1; i <= n; i++) {
    pre_sign[i] = pre_sign[i - 1];
    if(a[i] < 0)
      pre_sign[i] ^= 1;
    ones[i] = ones[i - 1];
```

```
    zeros[i] = zeros[i - 1];
    if(pre_sign[i])
      ones[i]++;
    else
      zeros[i]++;
  }
  ll ans = 0;
  for(int i = 1; i <= n; i++) {
    if(in_range(0))
      ans += n - next_zero[i] + 1;
    ll mul = 1;
    int j = i;
    bool overflow = 0;
    int iter = 0;
    while(j < next_zero[i] and !overflow) {
      iter++;
      //current mul will be same(except sign) until next_bigger[j + 1]
      if(abs((double) mul * a[j]) > 5e18)
        overflow = 1;
      mul *= a[j];
      int nxt = next_bigger[j + 1] - 1;
      nxt = min(nxt, next_zero[i] - 1);
      if(overflow)
        nxt = next_zero[i] - 1;
      if(!overflow) {
        if(in_range(abs(mul)))
          ans += pos_in(i, j, nxt);
        if(in_range(-abs(mul)))
          ans += neg_in(i, j, nxt);
      }
      else {
        if(r == 42)
          ans += pos_in(i, j, nxt);
        if(l == 42)
          ans += neg_in(i, j, nxt);
      }
      // printf("i = %d j = %d nxt = %d ans = %lld\n", i, j, nxt, ans);
      j = nxt + 1;
    }
    // printf("iter = %d\n", iter);
  }
  return ans;
}

int main() {
  // int id;
```

```
  // puts("id of test case:");
  // cin >> id;
  // char id_str[33];
  // sprintf(id_str, "%d", id);
  // string ids = id_str;
  // while(ids.size() < 2)
  //   ids = "0" + ids;
  // freopen(("input" + ids + ".txt").c_str(), "r", stdin);
  scanf("%d %lld %lld", &n, &l, &r);
  assert(1 <= n and n <= 100000);
  assert(-1e18 <= l and l <= 1e18);
  assert(-1e18 <= r and r <= 1e18);
  // assert(l != 42 and r != 42);
  for(int i = 1; i <= n; i++) {
    scanf("%lld", a + i);
    // assert(a[i] >= 0);
    assert(-1e9 <= a[i] and a[i] <= 1e9);
  }
  ll ans = calc();
  // ll brute_ans = brute();
  // printf("%lld %lld\n", ans, brute_ans);
  // assert(ans == brute_ans);
  cout << ans << endl;
  return 0;
}
```