Template:

```
/*  Anik Deb -> CSE,DUET
    Praying(Everything is depend on God):
    Hare Krishna Hare Krishna
    Krishna Krishna Hare Hare
    Hare Rama Hare Rama
    Rama Rama Hare Hare
*/
#include <bits/stdc++.h>
using namespace std;
#define fast ios::sync_with_stdio(false); cin.tie(0);
cin.exceptions(ios::badbit | ios::failbit);
#define precision(n) fixed<<setprecision(n)
#define lli long long int
#define ulli unsigned long long int
#define ld long double
#define max2(x,y) ((x>y)?x:y)
#define min2(x,y) ((x<y)?x:y)
#define inv(v) for(auto& i:v) cin>>i
#define outv(v) for(auto& i:v) cout<<i<<" "
#define pi acos(-1.0)
#define nline "\n"
#define vi vector<int>
#define vc vector<char>
#define p32 pair<int,int>
#define p64 pair<lli,lli>
#define caseOP(t,o) cout<<"Case "<<t<<": "<<o
#define caseOPi(t,o) printf("Case %d: %d\n",t,o)
#define caseOPl(t,o) printf("Case %d: %lld\n",t,o)
#define caseOPd(t,o) printf("Case %d: %lf\n",t,o)
#define cks(x) cout<<x<<nline;
#define ckd(x,y) cout<<x<<" "<<y<<nline;
#define ckt(x,y,z) cout<<x<<" "<<y<<" "<<z<<nline;
#define mod 1000000007
#define pb push_back
#define lcm(a,b) (a*b)/__gcd(a,b)
#define all(v) v.begin(),v.end()
char
alp[26]={'a','b','c','d','e','f','g','h','i','j','k','l','m','n','
o','p','q','r','s','t','u','v','w','x','y','z'};
const int MAX = 2e5+5;
const int INF = INT_MAX;

void solve(){
```

```cpp
}
int main()
{
    fast;
    int tt=1;
    //cin>>tt;
    for(int i=1;i<=tt;i++){
        solve();
    }
    return 0;
}
```

Print the Largest Sum Contiguous Subarray:O(n)

```cpp
void maxSubArraySum(int a[], int size)
{
    int max_so_far = INT_MIN, max_ending_here = 0,
        start = 0, end = 0, s = 0;

    for (int i = 0; i < size; i++) {
        max_ending_here += a[i];

        if (max_so_far < max_ending_here) {
            max_so_far = max_ending_here;
            start = s;
            end = i;
        }

        if (max_ending_here < 0) {
            max_ending_here = 0;
            s = i + 1;
        }
    }
    cout << "Maximum contiguous sum is " << max_so_far
         << endl;
    cout << "Starting index " << start << endl
         << "Ending index " << end << endl;
}
```

Seive:O(log(log(n)))

```cpp
#include<bits/stdc++.h>
using namespace std;
const int MAX=1e4+5;
int Prime[MAX],nPrime;
int mark[MAX]={0};

void seive(int n){
    int limit=sqrt(n)+2;
    nPrime=0;
    mark[1]=1;
    //all even are not prime
    for(int i=4;i<=n;i+=2) mark[i]=1;
    Prime[nPrime++]=2;

    //run loop for only odds
    for(int i=3;i<=n;i+=2){
        if(!mark[i]){
            Prime[nPrime++]=i;
            if(i<=limit){
                for(int j=i*i;j<=n;j+=(i*2)){
                    mark[j]=1;
                }
            }
        }

    }
}
int main()
{
    seive(10);
    for(int i=0;i<nPrime;i++) cout<<Prime[i]<<" ";
    cout<<mark[2]<<endl;
    return 0;
}
```

Segmented Seive: O(sqrt(n))

```
#include <bits/stdc++.h>
using namespace std;
void simpleSieve(int lmt, vector<int>& prime) {
   bool mark[lmt + 1];
   memset(mark, false, sizeof(mark));
   for (int i = 2; i <= lmt; ++i) {
      if (mark[i] == false) {
         prime.push_back(i);
         for (int j = i; j <= lmt; j += i)
            mark[j] = true;
      }
   }
}
void PrimeInRange(int low, int high) {
   int lmt = floor(sqrt(high)) + 1;
   vector<int> prime;
   simpleSieve(lmt, prime);
   int n = high - low + 1;
   bool mark[n + 1];
   memset(mark, false, sizeof(mark));
   for (int i = 0; i < prime.size(); i++) {
      int lowLim = floor(low / prime[i]) * prime[i];
      if (lowLim < low)
         lowLim += prime[i];
      for (int j = lowLim; j <= high; j += prime[i])
         mark[j - low] = true;
   }
   for (int i = low; i <= high; i++)
      if (!mark[i - low])
         cout << i << " ";
}
int main() {
   int low = 10, high = 50;
   PrimeInRange(low, high);
   return 0;
}
```

Prime Factorization: for each query (logn)

```
// C++ program to find prime factorization of a
// number n in O(Log n) time with precomputation
// allowed.
#include "bits/stdc++.h"
using namespace std;

#define MAXN 100001

// stores smallest prime factor for every number
int spf[MAXN];

// Calculating SPF (Smallest Prime Factor) for every
// number till MAXN.
// Time Complexity : O(nloglogn)
void sieve()
{
        spf[1] = 1;
        for (int i=2; i<MAXN; i++)

                // marking smallest prime factor for every
                // number to be itself.
                spf[i] = i;

        // separately marking spf for every even
        // number as 2
        for (int i=4; i<MAXN; i+=2)
                spf[i] = 2;

        for (int i=3; i*i<MAXN; i++)
        {
                // checking if i is prime
                if (spf[i] == i)
                {
                        // marking SPF for all numbers divisible by i
                        for (int j=i*i; j<MAXN; j+=i)

                                // marking spf[j] if it is not
                                // previously marked
                                if (spf[j]==j)
```

```
                              spf[j] = i;
                    }
          }
}

// A O(log n) function returning primefactorization
// by dividing by smallest prime factor at every step
vector<int> getFactorization(int x)
{
          vector<int> ret;
          while (x != 1)
          {
                    ret.push_back(spf[x]);
                    x = x / spf[x];
          }
          return ret;
}

// driver program for above function
int main(int argc, char const *argv[])
{
          // precalculating Smallest Prime Factor
          sieve();
          int x = 12246;
          cout << "prime factorization for " << x << " : ";

          // calling getFactorization function
          vector <int> p = getFactorization(x);

          for (int i=0; i<p.size(); i++)
                    cout << p[i] << " ";
          cout << endl;
          return 0;
}
```

Segment Tree with lazy:

```
const int MAX = 2e5+5;
int ara[MAX];
struct node{
   int sum,prop;
}segtree[MAX*4];
void build(int nodeNum,int segs,int sege){
   if(segs==sege){ //base case when segment lenth one the result of the segment is this element
      segtree[nodeNum].sum=ara[segs];
      return;
   }
```

```
    int mid=(segs+sege)/2;
    build(nodeNum*2,segs,mid);
    build(nodeNum*2+1,mid+1,sege);
    segtree[nodeNum].sum=segtree[nodeNum*2].sum+segtree[nodeNum*2+1].sum; //segment tree
for range minimum
    //segtree[nodeNum]=max(segtree[nodeNum*2],segtree[nodeNum*2+1]); segment tree for range
maximum
    //segtree[nodeNum]=segtree[nodeNum*2]+segtree[nodeNum*2+1]; segment tree for range sum

}
int query(int nodeNum,int segs,int sege,int qs,int qe,int carry=0){
    if(sege<qs || segs>qe) //completely outside ignore this segment
        return 0;
    if(segs>=qs && sege<=qe) //completely inside return result
        return segtree[nodeNum].sum+(sege-segs+1)*carry;
    //otherwise devide this segment by make recursion call
    int lnode=nodeNum<<1;
    int rnode=(nodeNum<<1)+1;
    int mid=(segs+sege)>>1;
    int left=query(lnode,segs,mid,qs,qe,carry+segtree[nodeNum].prop);
    int right=query(rnode,mid+1,sege,qs,qe,carry+segtree[nodeNum].prop);
    return left+right;
}
//point update
void update(int nodeNum,int segs,int sege,int qs,int qe,int x){
    if(sege<qs || segs>qe) //completely outside ignore this segment
        return;
    if(segs>=qs && sege<=qe) //completely inside return result
    {
        segtree[nodeNum].sum+=((sege-segs+1)*x);
        segtree[nodeNum].prop+=x;

        return;
    }
    //otherwise devide this segment by make recursion call
    int mid=(segs+sege)/2;
    update(nodeNum*2,segs,mid,qs,qe,x);
    update(nodeNum*2+1,mid+1,sege,qs,qe,x);
    segtree[nodeNum].sum=segtree[nodeNum*2].sum+segtree[nodeNum*2+1].sum+(sege-
segs+1)*segtree[nodeNum].prop;
}
void solve(){
    int n,q;cin>>n>>q;
    for(int i=1;i<=n;i++) cin>>ara[i];
    build(1,1,n);
//    for(int i=1;i<=2*n+1;i++) cout<<segtree[i].sum<<" ";cout<<nline;
//    for(int i=1;i<=2*n+1;i++) cout<<segtree[i].prop<<" ";cout<<nline;
    for(int i=1;i<=q;i++){
        int l,r,t;
        cin>>t;
        if(t==1){
            cin>>l>>r;
            cout<<query(1,1,n,l,r)<<nline;
        }
        else{
            cin>>l>>r;
            update(1,1,n,l,r,1);
        }
//        for(int i=1;i<=2*n+1;i++) cout<<segtree[i].sum<<" ";cout<<nline;
//        for(int i=1;i<=2*n+1;i++) cout<<segtree[i].prop<<" ";cout<<nline;
        cout<<nline;
    }
}
```

Given prime factorization and its number:
Print three integers modulo 109+7: the number, sum and product of the divisors.

Input:

```
2
2 2
3 1
```

Output:

```
6 28 1728
```

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;

const ll MOD = 1e9 + 7;

ll expo(ll base, ll pow) {
     ll ans = 1;
     while (pow) {
          if (pow & 1) ans = ans * base % MOD;
          base = base * base % MOD;
          pow >>= 1;
     }
     return ans;
}

ll p[100001], k[100001];
```

```
int main() {
     cin.tie(0)->sync_with_stdio(0);
     int n;
     cin >> n;
     for (int i = 0; i < n; i++) cin >> p[i] >> k[i];
     ll div_cnt = 1, div_sum = 1, div_prod = 1, div_cnt2 = 1;
     for (int i = 0; i < n; i++) {
          div_cnt = div_cnt * (k[i] + 1) % MOD;
          div_sum = div_sum * (expo(p[i], k[i] + 1) - 1) % MOD *
expo(p[i] - 1, MOD - 2) % MOD;
          div_prod = expo(div_prod, k[i] + 1) * expo(expo(p[i],
(k[i] * (k[i] + 1) / 2)), div_cnt2) % MOD;
          div_cnt2 = div_cnt2 * (k[i] + 1) % (MOD - 1);
     }
     cout << div_cnt << ' ' << div_sum << ' ' << div_prod;
     return 0;
}
```