#include<iostream>

#include<cstring>

#include<cstdio>

typedef long long LL;

typedef double D;

using namespace std;

const int L = 50004;

LL S[L], a[L], dp[L], Q[L], n, l;

D slope(int x, int y)

{

return (D)( (dp[x] + (a[x]+l)\*(a[x]+l) - dp[y] - (a[y]+l)\*(a[y]+l)) / ((a[x] - a[y]) \* 2.0) );

}

int main()

{

scanf("%lld%lld", &n, &l); l++;

S[0] = 0;

for (int s = 1; s <= n; s++)

{

scanf("%lld", &S[s]);

S[s] += S[s-1];

a[s] = S[s] + s;

}

memset(dp, 0, sizeof dp);

int top = 1, tail = 1;

memset(Q, 0, sizeof Q);

for (int s = 1; s <= n; s++)

{

while (top < tail && slope(Q[top], Q[top+1]) <= a[s]) top++;

int x = Q[top];

dp[s] = dp[x] + (a[s]-a[x]-l)\*(a[s]-a[x]-l);

while (top < tail && slope(Q[tail], s) < slope(Q[tail], Q[tail-1])) tail--;

Q[++tail] = s;

}

printf("%lld\n", dp[n]);

} //斜率优化

#include<algorithm>

#include<iostream>

#include<cstring>

#include<cstdio>

using namespace std;

const int L = 1000006;

int a[L], ind[L], tmp[L], tag[L], f[L][2], c[L];

int n, tot;

void uniq()

{

tot = 1, tag[0] = tmp[0];

int cur = tmp[0];

for (int s = 1; s < n; s++)

if (tmp[s] != cur)

tag[tot++] = tmp[s], cur = tmp[s];

for (int s = 0; s < n; s++)

{

int l = 0, r = tot-1;

while (l != r)

{

int m = (l+r) >> 1;

if (a[s] <= tag[m]) r = m;

else l = m+1;

}

ind[s] = l+1;

}

}

int lb(int k)

{

return k & -k;

}

void update(int k, int num)

{

for (int s = k; s <= n; s += lb(s)) c[s] = max(c[s], num);

}

int query(int k)

{

int ret = 0;

for (int s = k; s; s -= lb(s)) ret = max(ret, c[s]);

return ret;

}

int main()

{

scanf("%d", &n);

for (int s = 0; s < n; s++)

{

scanf("%d", &a[s]);

tmp[s] = a[s];

}

sort(tmp, tmp+n);

uniq();

/\*for (int s = 0; s < n; s++) printf("%d ", ind[s]);

printf("\n");\*/

memset(c, 0, sizeof c);

for (int s = 0; s < n; s++)

{

f[s][0] = query(ind[s]-1) + 1;

update(ind[s], f[s][0]);

}

memset(c, 0, sizeof c);

for (int s = n-1; s >= 0; s--)

{

f[s][1] = query(ind[s]-1) + 1;

update(ind[s], f[s][1]);

}

int ans = 0;

for (int s = 0; s < n; s++)

ans = max(ans, min(f[s][0], f[s][1]));

printf("%d\n", ans\*2 - 1);

} //树状数组LIS

typedef long long LL;

using namespace std;

const LL base = 1000000000LL;

const int L = 100005;

int n, m;

LL a[L];

struct node

{

node \*ch[2];

LL val, rnd, cnt, size;

node(LL x) {rnd = rand(), size = cnt = 1, ch[0] = ch[1] = NULL; val = x;}

int cmp(LL x)

{

if (x == val) return -1;

return (x > val ? 1 : 0);

}

void maintain()

{

size = cnt;

if (ch[0] != NULL) size += ch[0]->size;

if (ch[1] != NULL) size += ch[1]->size;

}

};

void rotate(node\* &k, int d)

{

node \*son = k->ch[d^1];

k->ch[d^1] = son->ch[d];

son->ch[d] = k;

k->maintain(), son->maintain();

k = son;

}

void insert(node\* &k, LL x)

{

if (k == NULL) k = new node(x);

else

{

int d = k->cmp(x);

if (d == -1) {k->cnt++;}

else

{

insert(k->ch[d], x);

if (k->ch[d]->rnd < k->rnd) rotate(k, d^1);

}

}

k->maintain();

}

void remove(node\* &k, LL x)

{

//printf("%lld %d %d %d\n", k->val, k->rnd, k->cnt, k->size);

int d = k->cmp(x);

if (d != -1) remove(k->ch[d], x);

else

{

if (k->cnt > 1) {k->cnt--;}

else

{

if (k->ch[0] != NULL && k->ch[1] != NULL)

{

int dd = (k->ch[0]->rnd < k->ch[1]->rnd ? 1 : 0);

rotate(k, dd);

remove(k->ch[dd], x);

}

else

{

if (k->ch[0] == NULL && k->ch[1] == NULL) k = NULL;

else if (k->ch[0] == NULL) k = k->ch[1];

else if (k->ch[1] == NULL) k = k->ch[0];

}

}

}

if (k != NULL) k->maintain();

}

LL rnk(node \*k, LL x)

{

LL ret = 0;

while (k != NULL)

{

int d = k->cmp(x);

if (k->ch[1] != NULL && d != 1) ret += k->ch[1]->size;

if (d == -1) return ret;

if (d == 0) ret += k->cnt;

k = k->ch[d];

}

}

// Treap

using namespace std;

const int L = 100005;

const int INF = 1 << 30;

int n, m, fa[L][17], dep[L], siz[L], gfa[L], dis[L], vis[L];

struct edge

{

int u, v, next;

} a[L<<1];

int head[L], tot = 0;

inline int rd() {}

void addedge(int u, int v) {}

void dfs(int x, int stp) {}

void getfa() {}

int LCA(int x, int y) {}

int gcrt = -1, totsize, grasub = 0;

void getroot(int x, int f)

{

siz[x] = 1;

int maxsub = 0;

for (int s = head[x]; ~s; s = a[s].next)

{

int v = a[s].v;

if (vis[v] || v == f) continue;

getroot(v, x), siz[x] += siz[v];

maxsub = max(maxsub, siz[v]);

}

maxsub = max(maxsub, totsize - siz[x]);

if (maxsub < grasub) gcrt = x, grasub = maxsub;

}

void build(int x, int f)

{

vis[x] = 1, gfa[x] = f;

for (int s = head[x]; ~s; s = a[s].next)

{

int v = a[s].v;

if (vis[v]) continue;

grasub = totsize = siz[v];

getroot(v, x);

build(gcrt, x);

}

}

int getdist(int x, int y)

{

int rt = LCA(x, y);

return dep[x] + dep[y] - (dep[rt] << 1);

}

void update(int x)

{

int u = x;

while (~x)

{

dis[x] = min(dis[x], getdist(x, u));

x = gfa[x];

}

}

int query(int x)

{

int ans = INF;

int u = x;

while (~x)

{

int cur = getdist(x, u) + dis[x];

ans = min(ans, cur);

x = gfa[x];

}

return ans;

}

totsize = n, grasub = n;

getroot(1, -1);

build(gcrt, -1);

update(1);

// 点分树

while (r < q[s].r) r++, add(ind[r]);

while (r > q[s].r) del(ind[r]), r--;

while (l < q[s].l) del(ind[l]), l++;

while (l > q[s].l) l--, add(ind[l]);

// 莫队

const D eps = 0.0000001;

using namespace std;

const int L = 1005;

const D DINF = 100000000.0;

int vis[L], pre[L], n;

D x[L], y[L], z[L], dis[L];

double hori(int a, int b) { return sqrt( (D)(x[a]-x[b])\*(x[a]-x[b]) + (D)(y[a]-y[b])\*(y[a]-y[b]) ); }

double vert(int a, int b) { return abs(z[a] - z[b]); }

D prim(D r)

{

memset(vis, 0, sizeof vis);

for (int s = 1; s <= n; s++)

{

dis[s] = vert(1, s) - r\*hori(1, s);

pre[s] = 1;

}

dis[1] = 0, vis[1] = 1;

D sumh = 0, sumv = 0;

for (int s = 1; s <= n; s++)

{

int v = 0; D minval = DINF;

for (int t = 1; t <= n; t++)

if (!vis[t] && dis[t] < minval)

minval = dis[t], v = t;

vis[v] = 1;

sumh += hori(pre[v], v);

sumv += vert(pre[v], v);

for (int t = 1; t <= n; t++)

{

D nw = vert(v, t) - r\*hori(v, t);

if (!vis[t] && dis[t] > nw)

dis[t] = nw, pre[t] = v;

}

}

return sumv / sumh;

}

void solve()

{

D r = 0, nr = DINF;

while (1)

{

nr = prim(r);

//printf("%.14lf %.14lf\n", r, nr);

if (abs(nr-r) < eps) break;

r = nr;

}

printf("%.3lf\n", r);

}

// 01分数规划

void toposort()

{

queue<int> Q;

memset(vis, 0, sizeof vis);

memset(dep, 0, sizeof dep);

for (int s = 1; s <= n; s++)

if (!vis[find(s)])

vis[find(s)] = 1, p[num++] = find(s);

for (int s = 0; s < num; s++)

if (!deg[p[s]])

Q.push(p[s]);

int pn = 0;

while (!Q.empty())

{

int u = Q.front(); Q.pop();

deg[u]--, pn++;

for (int s = head[u]; ~s; s = a[s].next)

{

int v = a[s].v;

deg[v]--;

dep[v] = max(dep[v], dep[u]+1);

if (!deg[v]) Q.push(v);

}

}

if (pn != num) { printf("-1\n"); return; }

int ans = 0;

for (int s = 0; s < num; s++) ans += (dep[p[s]]+1) \* cnt[p[s]];

printf("%d\n", ans);

}

// 拓扑序

struct anode

{

int id; LL len;

bool operator < (const anode x) const

{

return len + dist[id] > x.len + dist[x.id];

}

};

LL ans = 0;

void BFS(int x)

{

priority\_queue<anode> Q;

Q.push(anode{x, 0});

while (!Q.empty())

{

anode tmp = Q.top(); Q.pop();

int u = tmp.id;

LL len = tmp.len;

if (u == nd)

{

k--;

if (k == 0)

{

ans = len;

return;

}

}

for (int s = heada[u]; ~s; s = a[s].next)

{

int v = a[s].v;

Q.push(anode{v, len + a[s].w});

}

}

}

// k短路A\*

int dfn[L], low[L], vis[L], tm = 0, dep = 0;

int stk[L], top = 0, all = 0;

void tarjan(int x)

{

dfn[x] = low[x] = ++dep;

stk[++top] = x, vis[x] = 1;

for (int s = head[x]; ~s; s = a[s].next)

{

int v = a[s].v;

if (!dfn[v])

{

tarjan(v);

low[x] = min(low[x], low[v]);

}

else if (vis[v]) low[x] = min(low[x], dfn[v]);

}

int cur = 0;

if (low[x] == dfn[x])

{

all++;

while (cur != x)

{

cur = stk[top--];

pre[cur] = all;

val[all] += w[cur];

vis[cur] = 0;

}

}

}

// Tarjan

void dfs(int x)

{

stk[++top] = x;

//printf("# %d\n", x);

for (int &s = head[x]; ~s; s = a[s].next)

{

if (!a[s].vis)

{

a[s].vis = 1;

used++;

dfs(a[s].v);

break;

}

}

}

void fleury(int x)

{

stk[++top] = x;

while (top > 0)

{

int flg = 0;

int u = stk[top];

//printf("%d\n", u);

for (int &s = head[u]; ~s; s = a[s].next)

{

if (!a[s].vis)

{

flg = 1;

break;

}

}

if (!flg) ans[cnt++] = stk[top--];

else dfs(stk[top--]);

}

} // Fleury欧拉回路

int searchP()

{

queue<int> Q;

memset(da, -1, sizeof da);

memset(db, -1, sizeof db);

dis = INF;

for (int s = 1; s <= a; s++)

if (ma[s] == -1)

Q.push(s), da[s] = 0;

while (!Q.empty())

{

int u = Q.front(); Q.pop();

if (da[u] > dis) break;

for (int s = head[u]; ~s; s = e[s].next)

{

int v = e[s].v;

if (~db[v]) continue;

db[v] = da[u] + 1;

if (mb[v] == -1) dis = db[v];

else da[mb[v]] = db[v] + 1, Q.push(mb[v]);

}

}

return dis != INF;

}

int dfs(int u)

{

for (int s = head[u]; ~s; s = e[s].next)

{

int v = e[s].v;

if (!vis[v] && db[v] == da[u]+1)

{

vis[v] = 1;

if (mb[v] != -1 && db[v] == dis) continue;

if (mb[v] == -1 || dfs(mb[v]))

{

mb[v] = u, ma[u] = v;

return 1;

}

}

}

return 0;

}

int match()

{

int ans = 0;

memset(ma, -1, sizeof ma);

memset(mb, -1, sizeof mb);

while (searchP())

{

memset(vis, 0, sizeof vis);

for (int s = 1; s <= a; s++)

if (ma[s] == -1 && dfs(s))

ans++;

}

printf("%d\n", ans);

} // 最大匹配

void getConvexHull()

{

int top = 0;

for (int s = 0; s < n; s++)

{

while (top > 1 && cross(vec(ch[top-2], ch[top-1]), vec(ch[top-1], p[s])) < 0) top--;

ch[top++] = p[s];

}

int cur = top;

for (int s = n-2; s >= 0; s--)

{

while (top > cur && cross(vec(ch[top-1], ch[top-2]), vec(p[s], ch[top-1])) < 0) top--;

ch[top++] = p[s];

}

tot = n > 1 ? top-1 : top;

ch[tot] = ch[0];

}

void rotatingCalipers()

{

int ans = 0, top = 2 % tot;

for (int s = 0; s < tot; s++)

{

while (cross(vec(ch[s], ch[top]), vec(ch[s], ch[s+1])) > cross(vec(ch[s], ch[top+1]), vec(ch[s], ch[s+1]))) top = (top+1) % tot;

ans = max(ans, max(distSq(ch[s], ch[top]), distSq(ch[s+1], ch[top])));

}

printf("%d\n", ans);

} // 凸包+旋转卡壳

int bfs()

{

memset(dep, -1, sizeof dep);

queue<int> Q;

dep[0] = 0;

Q.push(0);

while (!Q.empty())

{

int u = Q.front(); Q.pop();

for (int s = head[u]; ~s; s = a[s].next)

{

int v = a[s].v;

if (dep[v] == -1 && a[s].w)

{

dep[v] = dep[u] + 1;

Q.push(v);

if (v == n+m+1) return 1;

}

}

}

return 0;

}

LL dfs(int x, LL cap)

{

if (x == n+m+1) return cap;

for (int &s = work[x]; ~s; s = a[s].next)

{

int v = a[s].v;

if (a[s].w && dep[v] == dep[x] + 1)

{

LL tmp = dfs(v, min(a[s].w, cap));

if (tmp)

{

a[s].w -= tmp;

a[s^1].w += tmp;

return tmp;

}

}

}

return 0;

}

LL dinic()

{

LL ret = 0, add;

while (bfs())

{

for (int s = 0; s <= m+n+1; s++) work[s] = head[s];

while (add = dfs(0, INF<<1)) ret += add;

}

return ret;

} // Dinic+最大权闭合子图

point joint(line e, line f)

{

D x = (e.B\*f.C - f.B\*e.C) / (e.A\*f.B - f.A\*e.B);

D y = (e.A\*f.C - f.A\*e.C) / (f.A\*e.B - e.A\*f.B);

return point{x, y};

}

point outerCentre(point E, point F, point G)

{

line H(E, F);

line I(F, G);

point M1 = mid(E, F);

point M2 = mid(F, G);

line X1(H.B, -H.A, H.A\*M1.y - H.B\*M1.x);

line X2(I.B, -I.A, I.A\*M2.y - I.B\*M2.x);

point C = joint(X1, X2);

return C;

}

void minCircleCover()

{

D curR = 0;

point CP = p[0];

for (int s = 1; s < n; s++)

{

if (dist(CP, p[s]) <= curR) continue;

CP = p[s], curR = 0;

for (int t = 0; t < s; t++)

{

if (dist(CP, p[t]) <= curR) continue;

CP = mid(p[s], p[t]);

curR = dist(p[s], p[t]) / 2;

for (int k = 0; k < t; k++)

{

if (dist(CP, p[k]) <= curR) continue;

CP = outerCentre(p[s], p[t], p[k]);

curR = dist(CP, p[k]);

}

}

}

printf("%.3lf\n", curR);

printf("%.3lf %.3lf\n", CP.x, CP.y);

} // 最小圆覆盖--增量法

void insert(int id)

{

int len = strlen(tmp), inc = 0;

for (int s = 0; s < len; s++)

{

int u = tmp[s] - 'a';

if (!tr[inc][u]) tr[inc][u] = tot++;

inc = tr[inc][u];

}

if (!nd[inc]) nd[inc] = id;

else pre[id] = nd[inc];

}

void getnext()

{

queue<int> Q;

f[0] = 0;

for (int s = 0; s < 26; s++)

{

int u = tr[0][s];

if (u) Q.push(u), lst[u] = 0, f[u] = 0;

}

while (!Q.empty())

{

int cur = Q.front(); Q.pop();

for (int s = 0; s < 26; s++)

{

int u = tr[cur][s];

if (!u) tr[cur][s] = tr[f[cur]][s];

else

{

int inc = f[cur];

while (inc && !tr[inc][s]) inc = f[inc];

f[u] = tr[inc][s];

lst[u] = nd[f[u]] ? f[u] : lst[f[u]];

Q.push(u);

}

}

}

}

void count(int x)

{

while (x)

{

cnt[nd[x]]++;

x = lst[x];

}

}

void match()

{

int len = strlen(ch), inc = 0;

for (int s = 0; s < len; s++)

{

inc = tr[inc][ch[s]-'a'];

if (nd[inc]) count(inc);

else if (lst[inc]) count(lst[inc]);

}

} // AC自动机

void getnext()

{

f[0] = f[1] = 0;

for (int s = 1; s < lent; s++)

{

int inc = f[s];

while (inc && tem[s] != tem[inc]) inc = f[inc];

f[s+1] = tem[s] == tem[inc] ? inc+1 : 0;

}

//for (int s = 0; s <= lent; s++) printf("%d ", f[s]);

}

void find()

{

int inc = 0;

for (int s = 0; s < lench; s++)

{

while (inc && ch[s] != tem[inc]) inc = f[inc];

if (ch[s] == tem[inc]) inc++;

if (inc == lent) printf("%d ", s-lent+2);

}

} // KMP

struct PAM

{

PAM()

{

memset(f, 0, sizeof f);

memset(ch, 0, sizeof ch);

}

int ch[L][26], f[L], len[L], lst, tot;

LL num[L], cnt[L];

int find(int s, int x)

{

while (S[s-len[x]-1] != S[s]) x = f[x];

return x;

}

void build()

{

memset(num, 0, sizeof num);

f[0] = 1, f[1] = 0;

len[0] = 0, len[1] = -1;

tot = 1, lst = 0;

for (int s = 1; s <= lenS; s++)

{

int u = S[s] - 'a';

int inc = find(s, lst);

if (!ch[inc][u])

{

len[++tot] = len[inc] + 2;

int pre = find(s, f[inc]);

f[tot] = ch[pre][u];

ch[inc][u] = tot;

num[tot] = num[f[tot]] + 1;

}

lst = ch[inc][u];

cnt[s] = num[lst];

}

}

} pam;

LL nex[L], ext[L];

void getnext()

{

nex[1] = lenT;

int inc = 1, mr = 2;

while (inc < lenT && T[inc] == T[inc+1]) inc++;

nex[2] = inc - 1;

for (int s = 3; s <= lenT; s++)

{

int p = mr + nex[mr] - 1, q = nex[s-mr+1];

if (q < p-s+1) nex[s] = q;

else

{

inc = max(p-s+1, 0);

while (s + inc <= lenT && T[s+inc] == T[inc+1]) inc++;

nex[s] = inc, mr = s;

}

}

}

void exkmp()

{

getnext();

int mr = 1, mlen = min(lenS, lenT);

while (mr <= mlen && S[mr] == T[mr]) mr++;

ext[1] = mr-1, mr = 1;

for (int s = 2; s <= lenS; s++)

{

int p = mr + ext[mr] - 1, q = nex[s-mr+1];

if (q < p-s+1) ext[s] = q;

else

{

int inc = max(p-s+1, 0);

while (s + inc <= lenS && inc+1 <= lenT && S[s+inc] == T[inc+1]) inc++;

ext[s] = inc, mr = s;

}

}

} // PAM+EXKMP

void fill()

{

int len = strlen(ch);

int inc = 0;

tg[inc++] = '[', tg[inc++] = '#';

for (int s = 0; s < len; s++) tg[inc++] = ch[s], tg[inc++] = '#';

tg[inc] = '\0';

}

int manacher()

{

int ans = 0;

fill();

int len = strlen(tg);

int mr = 0, inc = 0;

for (int s = 1; s <= len; s++)

{

if (mr > s) p[s] = min(mr - s, p[inc\*2 - s]);

else p[s] = 1;

while (tg[s - p[s]] == tg[s + p[s]]) p[s]++;

if (p[s] + s > mr) mr = p[s] + s, inc = s;

}

}

#include<iostream>

#include<cstring>

#include<cstdio>

#include<queue>

typedef long long LL;

using namespace std;

const int L = 1000;

char tmp[L];

int n, ch[L][26], f[L], tot = 1, ll;

LL dp[L][L], val[L];

void insert(int v)

{

int len = strlen(tmp), inc = 0;

for (int s = 0; s < len; s++)

{

int u = tmp[s] - 'a';

if (!ch[inc][u]) ch[inc][u] = tot++;

inc = ch[inc][u];

}

val[inc] += v;

}

void getnext()

{

queue<int> Q;

f[0] = 0;

for (int s = 0; s < 26; s++)

{

int u = ch[0][s];

if (u) Q.push(u), f[u] = 0;

}

while (!Q.empty())

{

int cur = Q.front(); Q.pop();

val[cur] += val[f[cur]];

for (int s = 0; s < 26; s++)

{

int u = ch[cur][s];

if (!u) ch[cur][s] = ch[f[cur]][s];

else

{

int inc = f[cur];

while (inc && !ch[inc][s]) inc = f[inc];

f[u] = ch[inc][s];

Q.push(u);

}

}

}

}

void init()

{

memset(f, 0, sizeof f);

memset(ch, 0, sizeof ch);

memset(val, 0, sizeof val);

}

int main()

{

scanf("%d", &n);

for (int s = 0; s < n; s++)

{

int v;

scanf("%s%d", tmp, &v);

insert(v);

}

scanf("%d", &ll);

getnext();

memset(dp, -1, sizeof dp);

dp[0][0] = 0;

LL ans = 0;

for (int s = 0; s <= ll; s++)

{

for (int t = 0; t < tot; t++)

{

if (dp[s][t] == -1) continue;

for (int k = 0; k < 26; k++)

{

int u = ch[t][k];

dp[s+1][u] = max(dp[s+1][u], dp[s][t] + val[u]);

}

}

}

for (int s = 0; s <= ll; s++)

for (int t = 0; t < tot; t++)

ans = max(ans, dp[s][t]);

printf("%lld\n", ans);

}

// SAM

#include<iostream>

#include<cstring>

#include<cstdio>

typedef long long LL;

typedef double D;

using namespace std;

const int L = 50004;

LL S[L], a[L], dp[L], Q[L], n, l;

D slope(int x, int y)

{

return (D)( (dp[x] + (a[x]+l)\*(a[x]+l) - dp[y] - (a[y]+l)\*(a[y]+l)) / ((a[x] - a[y]) \* 2.0) );

}

int main()

{

scanf("%lld%lld", &n, &l); l++;

S[0] = 0;

for (int s = 1; s <= n; s++)

{

scanf("%lld", &S[s]);

S[s] += S[s-1];

a[s] = S[s] + s;

}

memset(dp, 0, sizeof dp);

int top = 1, tail = 1;

memset(Q, 0, sizeof Q);

for (int s = 1; s <= n; s++)

{

while (top < tail && slope(Q[top], Q[top+1]) <= a[s]) top++;

int x = Q[top];

dp[s] = dp[x] + (a[s]-a[x]-l)\*(a[s]-a[x]-l);

while (top < tail && slope(Q[tail], s) < slope(Q[tail], Q[tail-1])) tail--;

Q[++tail] = s;

}

printf("%lld\n", dp[n]);

}