Computing fast average:

Prob:In this problem I have initially given an array .I have to print the average value of given interval;

Upadate:I have to change all the value of a interval with a (Value).

Idea:

When a node intersects I have passed the (lazy) in the left and right node.and did all the calculation for the root node.

(when I have the passed the lazy I set lazy of root with a invalid value.such that I can track that)

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

#define ll long long

struct st
{
    ll lazy,sum;
} tree[100005 * 8];

void init(ll node, ll b, ll e)
{
    // cout<<"haha"<<endl;
    if(b==e)
    {
        tree[node].sum = 0;
        tree[node].lazy = -1;
        return;
    }

    ll left = node * 2;
    ll right  = node * 2 + 1;
    ll mid  = (b + e) / 2;

    init(left,b, mid);
    init(right,mid+1,e);

    tree[node].sum = tree[left].sum + tree[right].sum;

}
void update(ll node, ll b, ll e,ll i, ll j, ll val)
{
    // cout<<"haha"<<endl;
    if(b>j || e<i)
    {
        if(tree[node].lazy == -1) return;
        else
        {
            tree[node].sum = (e - b + 1) * tree[node].lazy;
            return;
        }
    }

    if(b>=i && e<=j)
    {
        tree[node].lazy = val;
        tree[node].sum = ((e - b + 1) * val);
        return ;
    }

    ll left = node * 2;
    ll right  = node * 2 + 1;
    ll mid  = (b + e) / 2;

    if(tree[node].lazy!=-1)
    {
        tree[left].lazy =  tree[node].lazy;
        tree[right].lazy = tree[node].lazy;
        tree[node].lazy = -1;
    }

    update(left,b, mid,i,j,val);
    update(right,mid+1,e,i,j,val);

    tree[node].sum = tree[left].sum + tree[right].sum;
```

```
}

ll query(ll node, ll b, ll e, ll i, ll j, ll carry)
{
    // cout<<"hshs"<<endl;
    if(b>j || e<i)
    {
        return 0;
    }
    if(b>=i && e<=j)
    {
        if(carry == -1) return tree[node].sum;
        return (e - b + 1) * carry;

    }

    ll left = node * 2;
    ll right  = node * 2 + 1;
    ll mid  = (b + e) / 2;

    if(carry==-1)
        carry = tree[node].lazy;

    ll x = query(left,b, mid,i,j,carry);
    ll y = query(right,mid+ 1,e,i,j,carry);

    return x + y;
}
int main()
{
    ll t;
    scanf("%lld",&t);
    ll w = 0;
    while(t--)
    {
        ll N,Q;
        scanf("%lld %lld",&N,&Q);
        //memset(tree,0,sizeof(tree));
        init(1,0,N-1);

        printf("Case %lld:\n",++w);
        while(Q--)
        {
            // cout<<"iii"<<endl;
            ll type;
            scanf("%lld",&type);
            if(type == 1)
            {
                ll frm,to,vall;
                scanf("%lld %lld %lld",&frm,&to,&vall);
                update(1,0,N-1,frm,to,vall);
            }
            else
            {
                ll frm, to;
                scanf("%lld %lld",&frm,&to);
                ll ans = query(1,0,N-1,frm,to,-1);
                ///cout<<"ans = "<<ans<<endl;
                ll dif = (to - frm + 1);
                if(ans%dif == 0)
                    printf("%lld\n",ans/dif);
                else
                {
                    ll gcd = __gcd(dif,ans);
                    ans/=gcd;
                    dif/=gcd;
                    printf("%lld/%lld\n",ans,dif);
                }
            }
//            for(int i=1;i<=60;i++)
//            {
//                cout<<"node number "<<i<<" "<<" value
"<<tree[i].lazy<<endl;
//            }
        }
    }
    return 0;

}
```