```cpp
/*
  Segment Tree Lazy Propagation
  update_tree: Replace numbers [i, j] with value
  query_tree: Sum of numbers [i, j]
*/

LL arr[NN];
LL tree[NN*4], carry[NN*4];
bool visit[NN*4];

void build_tree(LL node, LL a, LL b)
{
   if(a > b) return;
   if(a == b)
   {
      tree[node] = arr[a];
      return;
   }
   LL left = node*2;
   LL right = left+1;
   LL mid = (a+b)/2;
   build_tree(left, a, mid);
   build_tree(right, mid+1, b);
   tree[node] = tree[left] + tree[right];
}

void update_node(LL node,LL a,LL b)
{
   tree[node] = (b-a+1)*carry[node];
   if(a != b)
   {
      LL left = node*2;
      LL right = left+1;
      carry[left] = carry[right] = carry[node];
      visit[left] = visit[right] = 1;
   }
   visit[node] = 0;
}

void update_tree(LL node,LL a,LL b,LL i,LL j,LL value)
{
   if(visit[node]) update_node(node,a,b);
   if(a > b || a > j || b < i) return;
   LL left = node*2;
   LL right = left+1;
   if(a >= i && b <= j)
   {
      tree[node] = (b-a+1)*value;
      if(a != b)
      {
         carry[left] = carry[right] = value;
         visit[left] = visit[right] = 1;
      }
      return;
   }
   LL mid = (a+b)/2;
   update_tree(left, a, mid, i, j, value);
   update_tree(right, mid+1, b, i, j, value);
   tree[node] = tree[left] + tree[right];
}

LL query_tree(LL node, LL a, LL b, LL i, LL j)
{
   if(a > b || a > j || b < i) return 0;
   if(visit[node]) update_node(node,a,b);
   if(a >= i && b <= j) return tree[node];
   LL left = node*2;
   LL right = left+1;
   LL mid = (a+b)/2;
   LL q1 = query_tree(left, a, mid, i, j);
   LL q2 = query_tree(right, mid+1, b, i, j);
   return q1+q2;
}

int main()
{
   int n = 5;
   for(int i = 1; i <= n; i++) arr[i] = i;
   mset(visit, 0);
   build_tree(1, 1, n);
   update_tree(1, 1, n, 1, 3, 5);
   cout << query_tree(1, 1, n, 2, 4) << endl;
}
```