#include<bits/stdc++.h>  
  
using namespace std ;  
  
const int N = 100020 ;  
  
int arr[N] , tree[4\*N] ;  
  
void Build(int node , int a , int b )  
{  
 if( a>b ) return;  
  
 if(a == b )  
 {  
 tree[node] = arr[b] ;  
 return ;  
 }  
  
 int left = (2 \*node) , right = left+1 , mid = (a+b)/2 ;  
  
 Build(left,a,mid) ;  
 Build(right,mid+1,b) ;  
  
 tree[node] = tree[left] + tree[right] ;  
  
}  
  
void Update(int node, int s , int e , int pos , int val )  
{  
 if ( s>pos || (e<pos) || (s>e) ) return; // base case  
  
 if( s == pos && e == pos )  
 {  
 tree[node] += val ; // update  
 return ;  
 }  
  
 int left = node\*2 , right = left+1 , mid = (s+e)/2 ;  
  
 Update(left,s,mid,pos,val) ;  
 Update(right,mid+1,e,pos,val) ;  
  
 tree[node] = tree[left] + tree[right] ; // after update it update each node again  
  
  
}  
  
int query(int node, int st, int en , int a, int b )  
{  
 if( a > en || b < st || en < st ) return 0;  
 if( a <= st && b >= en ) return tree[node];  
  
 int left = 2\*node , right = left+1 , mid = (st+en)/2 ;  
  
 int q1 = query(left, st , mid , a, b ) ;  
 int q2 = query(right , mid+1 , en , a, b) ;  
  
 return q1+q2 ;  
}  
  
  
  
  
  
int main()  
{  
  
 int t ,tcase ,n,q ;  
  
 scanf("%d",&t) ;  
  
 tcase =1 ;  
 while(t--)  
 {  
 scanf("%d %d",&n,&q) ;  
 memset(arr,0,sizeof(arr)) ;  
 for(int i=1 ; i<=n ; i++)  
 {  
 scanf("%d",&arr[i]) ;  
 }  
 memset( tree , 0 , sizeof( tree ) );  
 Build(1,1,n);  
  
 printf("Case %d:\n",tcase++) ;  
 int op ;  
 while(q--)  
 {  
 scanf("%d",&op) ;  
  
 if( op ==1)  
 {  
 int pos ;  
 scanf("%d",&pos) ;  
 pos++ ;  
  
 int val = arr[pos] ;  
  
 printf("%d\n",val) ;  
 Update(1,1,n,pos,-val) ;  
  
 arr[pos] = 0;  
 }  
  
 else if( op == 2) // add money  
 {  
 int pos , val ;  
 scanf("%d %d",&pos,&val) ;  
  
 pos++ ;  
 arr[pos] += val ;  
  
 Update(1,1,n , pos , val) ;  
  
 }  
 else  
 {  
 int x , y ;  
 scanf("%d %d",&x,&y) ;  
  
 ++x , ++y ;  
 int res = query(1,1,n,x,y) ;  
 printf("%d\n",res) ;  
 }  
 }  
  
 }  
  
 return 0 ;  
}