```
#include <stdio.h>
__global__ void add_arrays_gpu(float *in1,float *in2,float *out)
{
	int idx=threadIdx.x;
	out[idx]=in1[idx]+in2[idx];
}
int main()
{
	// pointers to host memory
	float *a,*b,*c;
	// pointers to device memory
	float *a_d,*b_d,*c_d;
	int N=18;
	int i;

	// allocate arrays a, b and c on host
	a=(float*)malloc(N*sizeof(float));
	b=(float*)malloc(N*sizeof(float));
	c=(float*)malloc(N*sizeof(float));

	// allocate arrays a_d, b_d and c_d on device
	cudaMalloc((void**)&a_d,sizeof(float)*N);
	cudaMalloc((void**)&b_d,sizeof(float)*N);
	cudaMalloc((void**)&c_d,sizeof(float)*N);

	// initialize arrays a and b
	for(i=0;i<N;i++){
		a[i]= (float) i*i;
		b[i]=-(float) i/2.0f;
	}
	// execution configuration: How the threads are arranged, FLAT and LINEAR.
	dim3 dimGrid(1),dimBlock(N);
	add_arrays_gpu<<<dimGrid,dimBlock>>>(a_d, b_d, c_d);

	cudeMemcpy(a_d,a,sizeof(long)*N,cudeMemcpyDeviceToHost);
	cudaMemcpy(b_d,b,sizeof(float)*N,cudaMemcpyHostToDevice);
	int k;
	for(k = e-1; k>=0;k--)
	{
		N/=2;
		dim3 dimGrid(1),dimBlock(N);
		add_gpu<<<dimGrid,dimBlock>>>(a_d,c_d);
		if(k)
		{
			b_d = a_d;
			a_d = c_d;
			c_d = b_d;
		}
	}
	cudeMemcpy(c,c_d,sizeof(long)*N,cudeMemcpyDeviceToHost);

	for(i=0;i<N;i++)
		printf("c[%d]=%f\n",i,c[i]);
	print("%i, c[0]); // <-sum
	// Free stuff
}
```