Week5 > q1_1.cu > ...

```cuda
#include <stdio.h>
#include <cuda_runtime.h>

__global__ void vectorAdd(float *A, float *B, float *C, int N) {
    int i = threadIdx.x;
    if (i < N) {
        C[i] = A[i] + B[i];
    }
}

int main() {
    int N = 8;
    size_t size = N * sizeof(float);

    float *h_A, *h_B, *h_C;
    float *d_A, *d_B, *d_C;

    h_A = (float*)malloc(size);
    h_B = (float*)malloc(size);
    h_C = (float*)malloc(size);

    for (int i = 0; i < N; i++) {
        h_A[i] = i;
        h_B[i] = i * 2;
    }

    cudaMalloc(&d_A, size);
    cudaMalloc(&d_B, size);
    cudaMalloc(&d_C, size);

    cudaMemcpy(d_A, h_A, size, cudaMemcpyHostToDevice);
    cudaMemcpy(d_B, h_B, size, cudaMemcpyHostToDevice);

    vectorAdd<<<1, N>>>(d_A, d_B, d_C, N);

    cudaMemcpy(h_C, d_C, size, cudaMemcpyDeviceToHost);

    for (int i = 0; i < N; i++) {
        printf("%f ", h_C[i]);
    }

    cudaFree(d_A);
    cudaFree(d_B);
    cudaFree(d_C);
    free(h_A);
    free(h_B);
    free(h_C);

    return 0;
}
```

Week5 > q1_2.cu > ...

```cuda
#include <stdio.h>
#include <cuda_runtime.h>

__global__ void vectorAdd(float *A, float *B, float *C, int N) {
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    if (i < N) {
        C[i] = A[i] + B[i];
    }
}

int main() {
    int N = 1024;
    size_t size = N * sizeof(float);

    float *h_A, *h_B, *h_C;
    float *d_A, *d_B, *d_C;

    h_A = (float*)malloc(size);
    h_B = (float*)malloc(size);
    h_C = (float*)malloc(size);

    for (int i = 0; i < N; i++) {
        h_A[i] = i;
        h_B[i] = i * 2;
    }

    cudaMalloc(&d_A, size);
    cudaMalloc(&d_B, size);
    cudaMalloc(&d_C, size);

    cudaMemcpy(d_A, h_A, size, cudaMemcpyHostToDevice);
    cudaMemcpy(d_B, h_B, size, cudaMemcpyHostToDevice);

    int threadsPerBlock = 256;
    int blocks = (N + threadsPerBlock - 1) / threadsPerBlock;

    vectorAdd<<<blocks, threadsPerBlock>>>(d_A, d_B, d_C, N);

    cudaMemcpy(h_C, d_C, size, cudaMemcpyDeviceToHost);

    for (int i = 0; i < 10; i++) {
        printf("%f ", h_C[i]);
    }

    cudaFree(d_A);
    cudaFree(d_B);
    cudaFree(d_C);
    free(h_A);
    free(h_B);
    free(h_C);

    return 0;
}
```

```
(dse) mca@computinglab25-22:~/Desktop/PPL_230962278/Week5$ nvcc q1_1.cu -o q1
(dse) mca@computinglab25-22:~/Desktop/PPL_230962278/Week5$ ./q1
0.000000 3.000000 6.000000 9.000000 12.000000 15.000000 18.000000 21.000000 (dse) mca@computinglab25-22:~/Desktop/PPL_230962278/Week5$ nvcc q1_2.cu -o q1_2
(dse) mca@computinglab25-22:~/Desktop/PPL_230962278/Week5$ ./q1_2
0.000000 3.000000 6.000000 9.000000 12.000000 15.000000 18.000000 21.000000 24.000000 27.000000 (dse) mca@computinglab25-22:~/Desktop/PPL_230962278/Week5$ ^C
(dse) mca@computinglab25-22:~/Desktop/PPL_230962278/Week5$
```

q2.cu q3.cu

Week5 > q2.cu > ...

```cuda
#include <stdio.h>
#include <stdlib.h>
#include <cuda_runtime.h>

__global__ void vectorAdd(float *A, float *B, float *C, int N) {
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    if (i < N) {
        C[i] = A[i] + B[i];
    }
}

int main() {
    int N;
    scanf("%d", &N);

    size_t size = N * sizeof(float);

    float *h_A = (float*)malloc(size);
    float *h_B = (float*)malloc(size);
    float *h_C = (float*)malloc(size);

    for (int i = 0; i < N; i++) {
        h_A[i] = i;
        h_B[i] = i * 2;
    }

    float *d_A, *d_B, *d_C;

    cudaMalloc(&d_A, size);
    cudaMalloc(&d_B, size);
    cudaMalloc(&d_C, size);

    cudaMemcpy(d_A, h_A, size, cudaMemcpyHostToDevice);
    cudaMemcpy(d_B, h_B, size, cudaMemcpyHostToDevice);

    int threadsPerBlock = 256;
    int blocks = (N + threadsPerBlock - 1) / threadsPerBlock;

    vectorAdd<<<blocks, threadsPerBlock>>>(d_A, d_B, d_C, N);

    cudaMemcpy(h_C, d_C, size, cudaMemcpyDeviceToHost);

    for (int i = 0; i < N; i++) {
        printf("%f ", h_C[i]);
    }

    cudaFree(d_A);
    cudaFree(d_B);
    cudaFree(d_C);

    free(h_A);
    free(h_B);
    free(h_C);

    return 0;
}
```

```
(dse) mca@computinglab25-22:~/Desktop/PPL_230962278/Week5$ nvcc q2.cu -o q2
(dse) mca@computinglab25-22:~/Desktop/PPL_230962278/Week5$ ./q2
20
0.000000 3.000000 6.000000 9.000000 12.000000 15.000000 18.000000 21.000000 24.000000 27.000000 30.000000 33.000000 36.000000 39.000000 42.000000 45.000000 48.000000 51.00000
```

q3.cu

Week5 > q3.cu > main()

```cuda
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <cuda_runtime.h>

__global__ void computeSine(float *angles, float *output, int N) {
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    if (i < N) {
        output[i] = sinf(angles[i]);
    }
}

int main() {
    int N;
    scanf("%d", &N);

    size_t size = N * sizeof(float);

    float *h_angles = (float*)malloc(size);
    float *h_output = (float*)malloc(size);

    for (int i = 0; i < N; i++) {
        scanf("%f", &h_angles[i]);
    }

    float *d_angles, *d_output;

    cudaMalloc(&d_angles, size);
    cudaMalloc(&d_output, size);

    cudaMemcpy(d_angles, h_angles, size, cudaMemcpyHostToDevice);

    int threadsPerBlock = 256;
    int blocks = (N + threadsPerBlock - 1) / threadsPerBlock;

    computeSine<<<blocks, threadsPerBlock>>>(d_angles, d_output, N);

    cudaMemcpy(h_output, d_output, size, cudaMemcpyDeviceToHost);

    for (int i = 0; i < N; i++) {
        printf("%f ", h_output[i]);
    }

    cudaFree(d_angles);
    cudaFree(d_output);

    free(h_angles);
    free(h_output);

    return 0;
}
```

```
(dse) mca@computinglab25-22:~/Desktop/PPL_230962278/Week5$ nvcc q3.cu -o q3
(dse) mca@computinglab25-22:~/Desktop/PPL_230962278/Week5$ ./q3
4
0
1.5708
3.1416
4.7124
0.000000 1.000000 -0.000007 -1.000000 (dse) mca@computinglab25-22:~/D
```