**GPA CIE 2**

**Neeti Kurulkar**

**BE A Roll No: 31040**

**Q.1.**

**#include <stdio.h>**

**#include <cuda\_runtime.h>**

**// Macro for error checking**

**#define cudaCheckError(ans) { gpuAssert((ans), \_\_FILE\_\_, \_\_LINE\_\_); }**

**inline void gpuAssert(cudaError\_t code, const char \*file, int line, bool abort=true)**

**{**

**if (code != cudaSuccess)**

**{**

**fprintf(stderr,"GPUassert: %s %s %d\n", cudaGetErrorString(code), file, line);**

**if (abort) exit(code);**

**}**

**}**

**// Kernel: each thread adds one element**

**\_\_global\_\_ void vectorAdd(int \*A, int \*B, int \*C, int N) {**

**int idx = blockDim.x \* blockIdx.x + threadIdx.x;**

**if (idx < N) {**

**C[idx] = A[idx] + B[idx];**

**}**

**}**

**int main() {**

**int N = 10;**

**int size = N \* sizeof(int);**

**int h\_A[N], h\_B[N], h\_C[N];**

**for (int i = 0; i < N; i++) {**

**h\_A[i] = i;**

**h\_B[i] = i \* 2;**

**}**

**int \*d\_A, \*d\_B, \*d\_C;**

**cudaCheckError(cudaMalloc((void\*\*)&d\_A, size));**

**cudaCheckError(cudaMalloc((void\*\*)&d\_B, size));**

**cudaCheckError(cudaMalloc((void\*\*)&d\_C, size));**

**cudaCheckError(cudaMemcpy(d\_A, h\_A, size, cudaMemcpyHostToDevice));**

**cudaCheckError(cudaMemcpy(d\_B, h\_B, size, cudaMemcpyHostToDevice));**

**int threadsPerBlock = 256;**

**int blocksPerGrid = (N + threadsPerBlock - 1) / threadsPerBlock;**

**vectorAdd<<<blocksPerGrid, threadsPerBlock>>>(d\_A, d\_B, d\_C, N);**

**cudaCheckError(cudaGetLastError()); // check launch**

**cudaCheckError(cudaDeviceSynchronize()); // check runtime**

**cudaCheckError(cudaMemcpy(h\_C, d\_C, size, cudaMemcpyDeviceToHost));**

**printf("Result:\n");**

**for (int i = 0; i < N; i++) {**

**printf("%d + %d = %d\n", h\_A[i], h\_B[i], h\_C[i]);**

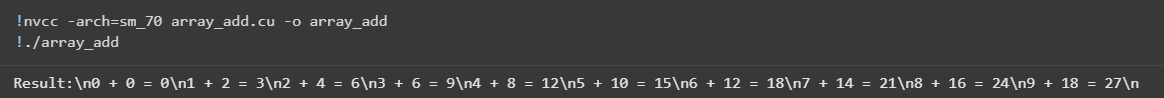
**}**

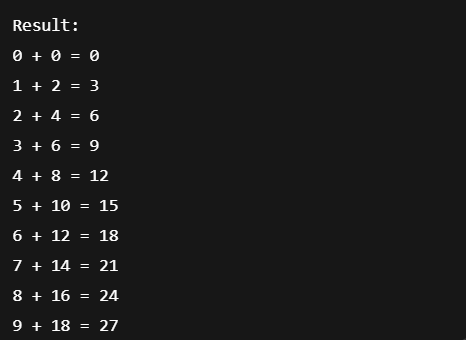
**cudaFree(d\_A); cudaFree(d\_B); cudaFree(d\_C);**

**return 0;**

**}**

**Output:**

****

****

**Q.2.**

**#include <stdio.h>**

**#include <CL/cl.h>**

**const char \*kernelSource =**

**"\_\_kernel void printNameDay() { \n"**

**" int id = get\_global\_id(0); \n"**

**" printf(\"Global ID %d: Neeti - Tuesday\\n\", id); \n"**

**"} \n";**

**int main() {**

**cl\_platform\_id platform;**

**cl\_device\_id device;**

**cl\_context context;**

**cl\_command\_queue queue;**

**cl\_program program;**

**cl\_kernel kernel;**

**// 1. Get platform and device**

**clGetPlatformIDs(1, &platform, NULL);**

**clGetDeviceIDs(platform, CL\_DEVICE\_TYPE\_ALL, 1, &device, NULL);**

**// 2. Create context and command queue**

**context = clCreateContext(0, 1, &device, NULL, NULL, NULL);**

**queue = clCreateCommandQueue(context, device, 0, NULL);**

**// 3. Build program**

**program = clCreateProgramWithSource(context, 1, &kernelSource, NULL, NULL);**

**clBuildProgram(program, 0, NULL, NULL, NULL, NULL);**

**// 4. Create kernel**

**kernel = clCreateKernel(program, "printNameDay", NULL);**

**// 5. Run kernel (5 work-items)**

**size\_t globalSize = 5;**

**clEnqueueNDRangeKernel(queue, kernel, 1, NULL, &globalSize, NULL, 0, NULL, NULL);**

**clFinish(queue);**

**// 6. Cleanup**

**clReleaseKernel(kernel);**

**clReleaseProgram(program);**

**clReleaseCommandQueue(queue);**

**clReleaseContext(context);**

**return 0;**

**}**

**Output:**

**A screenshot of a computer program

AI-generated content may be incorrect.**