**Question1**.

**Part1**:  
Compile Code:  
gcc -o q1p1 q1p1.c -lm

./q1p1

Analysis:

Particle amount is hard coded to 1024.

2-D array, [N][0] stores x, [N][1] stores y.

Core functions:   
double computeDistance(double particles[N][2], int i, int j) double findMinimumDistance(double particles[N][2])

Output:

A black background with white text

Description automatically generated

**Part2:**  
Compile Code:  
nvcc -arch=sm\_60 -O2 q1p2.cu

nvprof ./a.out

Analysis:

In order to simplify the logic, data structure change to:  
 particle[INDEX]

INDEX % 2 == 0 : x

INDEX % 2 == 1 : y

Core functions:

findMinimumDistance(float \*particles, float \*minDistance)

Output:

A screenshot of a computer

Description automatically generated

**Part3:**  
Compile Code:  
nvcc -arch=sm\_60 -O2 q1p3.cu

nvprof ./a.out

Analysis:

Same data structure as part2.

Travel array, but only calculates the target pair.

Core functions:

findMinimumDistance(float \*particles, float \*minDistance)

Output:

**A screenshot of a computer

Description automatically generated**

**Part 4 discussion:**  
Following points could be taken into account.

- Share Memory Usage  
- Optimize Reduction Algorithm

- Optimize traveling Algorithm

- If we have enough memory , we can create a new array to store all possibilities, the call GPU \_\_global\_\_ , that looks like bigO =1

**Question2**.