```
/*
 * This is a program to implement Odd-Even Sort using CUDA * It uses N/2 threads
*/
#include<cuda.h>
#include<stdio.h>
#include<time.h>
#include<stdlib.h>
#define N 1024

__global__ void oddeven(int *a, int flag)
{
 int index = blockIdx.x * blockDim.x + threadIdx.x;
  int temp;
  if((index >= N/2 - 1) && flag % 2 != 0) return;

  if(flag % 2 == 0) //if even phase
  {
    if(a[index *2 ] > a[index * 2 + 1])
    {
      temp = a[index * 2];
      a[index * 2] = a[index *2 +1];
      a[index * 2 +1] = temp;
    }
  }
  else { //if odd phase
    if(a[index * 2 +1 ] > a[index *2 + 2])
  {
      temp = a[index * 2 + 1];
      a[index * 2 + 1] = a[index*2+2];
      a[index*2+2] = temp;
    }
  }
}

int main()
{
 int *a;
  int *deva;
  int i;
  int size = sizeof(int) * N;
  srand((unsigned)time(NULL));

  //allocate memory in host
 a = (int *)malloc(size);

  //allocate memory in CUDA (device) memory
  cudaMalloc((void **)&deva, size);

  //puting some random values in memory for generating data for sorting
  for(i=0;i<N;i++)
  {
    a[i] = rand()%N;
  }

 printf("\nNumbers before sorting: ");
  for(i=0;i<N;i++)
   {
     printf("%d ", a[i]);
   }

//recording starting time
 double start_time = clock();

 //copy host memory data in CUDA (device) memory
```

```
  cudaMemcpy(deva, a, size, cudaMemcpyHostToDevice);

  // launch a kernel N-1 times for Odd-even sort
  for(i=0;i<N;i++)
  {
    oddeven<<<N/1024, 512>>>(deva, i); //512 threads per block and total N/2/512
blocks
   }

  //copy the result back into host memory
  cudaMemcpy(a, deva, size, cudaMemcpyDeviceToHost);

  //Lets see the execution time
  printf("\nExecution time : %lf seconds", (clock()-start_time)/CLOCKS_PER_SEC);

  //print the result
  printf("\nOutput: ");
  for(i=0;i<N;i++)
  {
    printf("%d ", a[i]);
  }
  return 0;
}
```