```
#include <stdio.h>
#include <float.h>
#include <unistd.h>
#include <pwd.h>

#define N 40
#define THREADS 10

__global__ void findmax(float *A, float *max)
{
   __shared__ float smax[THREADS];
   int i = blockDim.x*blockIdx.x+threadIdx.x, half;

   if (i < N)
      smax[threadIdx.x] = A[i];
   else
      smax[threadIdx.x] = -FLT_MAX;

   half = 1<<(int)(log2((float)(blockDim.x-1)));
   for(unsigned int s=half; s>0; s>>=1)
   {
      if (threadIdx.x+s < blockDim.x)
         if (threadIdx.x < s)
            if (smax[threadIdx.x] < smax[threadIdx.x+s])
               smax[threadIdx.x] = smax[threadIdx.x+s];
      __syncthreads();
   }
   if (threadIdx.x == 0)
      max[blockIdx.x] = smax[threadIdx.x];
}

int main()
{
   float A[N], *A_d, *max_arr, *max_arr_d, max;
   int i, dev;

   dim3 dimBlock(THREADS);
   dim3 dimGrid((N+dimBlock.x-1)/dimBlock.x);

   dev = (getpwuid(getuid())->pw_name[3]-'0')%2? 1: 0;
   cudaSetDevice(dev);

   srand(1);
   for (i=0; i<N; i++) {
      A[i] = rand() % 999;
      printf("%2.1f ", A[i]);
   }
   printf("\n");

   cudaMalloc((void **) &A_d, sizeof(float)*N);
   cudaMemcpy(A_d, A, sizeof(float)*N, cudaMemcpyHostToDevice);

   cudaMalloc((void **) &max_arr_d, dimGrid.x*sizeof(float));

   findmax<<<dimGrid, dimBlock>>>(A_d, max_arr_d);

   max_arr = (float*)malloc(dimGrid.x*sizeof(float));
   cudaMemcpy(max_arr, max_arr_d, dimGrid.x*sizeof(float), cudaMemcpyDeviceToHost);

   // find the maximum
   max = max_arr[0];
   for (i=1; i<dimGrid.x; i++)
      if (max < max_arr[i]) max = max_arr[i];

   printf("%f\n", max);

   cudaFree(A_d);
   cudaFree(max_arr_d);
   free(max_arr);
   exit(0);
}
```