```
// includes, system
#include <stdio.h>
#include <assert.h>

// Simple utility function to check for CUDA runtime errors
void checkCUDAError(const char *msg);

////////////////////////////////////////////////////////////////////////////////
//////
// Program main
////////////////////////////////////////////////////////////////////////////////
//////
int main( int argc, char** argv)
{
    // pointer and dimension for host memory
    int n, dimA;
    float *h_a;

    // pointers for device memory
    float *d_a, *d_b;

    // allocate and initialize host memory
    // Bonus: try using cudaMallocHost in place of malloc
    dimA = 8;
    h_a = (float *) malloc(dimA*sizeof(float));
    for (n=0; n<dimA; n++)
    {
        h_a[n] = (float) n;
    }

    // Part 1 of 5: allocate device memory
    size_t memSize = dimA*sizeof(float);
    cudaMalloc( (void**)&d_a, memSize );
    cudaMalloc( (void**)&d_b, memSize );

    // Part 2 of 5: host to device memory copy
    cudaMemcpy( d_a, h_a, memSize, cudaMemcpyHostToDevice   );

    // Part 3 of 5: device to device memory copy
    cudaMemcpy( d_b, d_a, memSize, cudaMemcpyDeviceToDevice );

    // clear host memory
    for (n=0; n<dimA; n++)
    {
        h_a[n] = 0.f;
    }

    // Part 4 of 5: device to host copy
    cudaMemcpy( h_a, d_b, memSize, cudaMemcpyDeviceToHost );

    // Check for any CUDA errors
    checkCUDAError("cudaMemcpy calls");

    // verify the data on the host is correct
    for (n=0; n<dimA; n++)
    {
        assert(h_a[n] == (float) n);
    }

    // Part 5 of 5: free device memory pointers d_a and d_b
```

```
    cudaFree( d_b );
    cudaFree( d_a );

    // Check for any CUDA errors
    checkCUDAError("cudaFree");

    // free host memory pointer h_a
    // Bonus: be sure to use cudaFreeHost for memory allocated with
cudaMallocHost
    free(h_a);

    // If the program makes it this far, then the results are correct and
    // there are no run-time errors.  Good work!
    printf("Correct!\n");

    return 0;
}

void checkCUDAError(const char *msg)
{
    cudaError_t err = cudaGetLastError();
    if( cudaSuccess != err)
    {
        fprintf(stderr, "Cuda error: %s: %s.\n", msg, cudaGetErrorString(
err) );
        exit(-1);
    }
}
```