CSC8014 Topics: GPU Programming
Student: Lina Mi @377283

Homework #2

(Since we are going to make heavy use of pointers and memory management, the first 4 problems are related to low level C language)

1. Creates two arrays of integers of size 100 and sets 45th element in each array to 34.

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int *a, *b;
   a=(int*)malloc(100*sizeof(int));
   b=(int*)malloc(100*sizeof(int));
   a[44]=34;
   b[44]=34;
   return 0;
}
```

2. Creates two arrays of floats of size 10 and sets 5th element in each array to 12.34.

```
#include <stdio.h>
#include <stdlib.h>
int main(){
     float *a, *b;
     a=(float *) malloc(10*sizeof(float));
     b=(float*) malloc(10*sizeof(float));
     *(a+4)=12.34;
     *(b+4)=12.34;
    return 0;
}
```

3. What is the output of the following code:

```
#include <stdio.h>
int main(void) {
    int *x;
    x = (int*)malloc(10 * sizeof(int));
    for (int i = 0; i < 10; i++)
            *(x + i) = i;
    for (int i = 0; i < 10; i++)
            printf("%d", x[i]);
    return 0;
}
```

Ans: the output of the code is 0123456789

4. The following C code is supposed to create two integer pointers x and y and set the values to which they point to 21 and 99, respectively. Please identify if the code is correct and if it's not, fix it!

```
void method(){
   int* x,y;
   x = (int*) malloc(sizeof(int));
   y = (int*) malloc(sizeof(int));

   if (!x && b)){
      printf("Out of memory");
      exit(-1);
   }
   *x = 21;
   *y = 99;
}
```

Ans: 1) the code need to include stdio.h and stdlib.h head files;

2) there is no main function

3) y should be integer pointer, not integer

3) the condition should be !x || !y

The code is as following:

```
#include <stdio.h>
#include<stdlib.h>
int main(void){
   int *x, *y;
   x=(*int) malloc(sizeof(int));
   y=(*int) malloc(sizeof(int));
   if(!x || !y){
      printf("out of memory!");
      exit(-1);
    }
   *x=21;
   *y=99;
 }
```

5. Assume the maximum number of blocks in a single launch is 65,535 and the maximum number of threads per block is 128. Suppose you would like to use the following kernel method "add()" for operations on an arrays a, b, and c of N = 50,000 elements.

```
__global__ void add( int *a, int *b, int *c ) {
   int tid = blockIdx.x;
   if (tid < N)
      c[tid] = a[tid] + b[tid];
}
```

In the main program, which of the following kernel launch is correct? If it's wrong, how to correct it? (if the kernel method needs to be modified, please do so)

a. add<<<N, 1>>>(dev_a, dev_b, dev_c);

b. add<<<1, N>>>(dev_a, dev_b, dev_c);

c. add<<<41, 128>>>(dev_a, dev_b, dev_c);

Ans: choice a is correct. Since in the kernel method, the index of array is only depend on block index, no thread index involved and the size of array N=50,000 is less than maximum number of blocks in single launch of 65,535, so the kernel can be launched with 50,000 blocks and one thread for each block.

6. Repeat problem 5 for N = 10,000,000. (Note: N is greater than 65,535 x 128 = 8,388,480)

   Do we need to modify the kernel method "add"? How?

   Ans: since the N=10,000,000 is greater than 66,635(maximum block per launch)*128(maximum thread per block), the kernel can not be launched at single time, it is need to modify both kernel code and code to launch kernel:

```
__global__  void add(*a, *b,*c){
   tid= threadIdx.x + blockIdx.x * blockDim.x;
   while(tid<N){
       c[tid]=a[tid]+b[tid];
       tid+=blockDim.x * gridDim.x;
  }
```

   In the main function, the code to launch kernel:

```
add<<<128,128>>>(dev_a, dev_b, dev_c);
```