Answer: 1

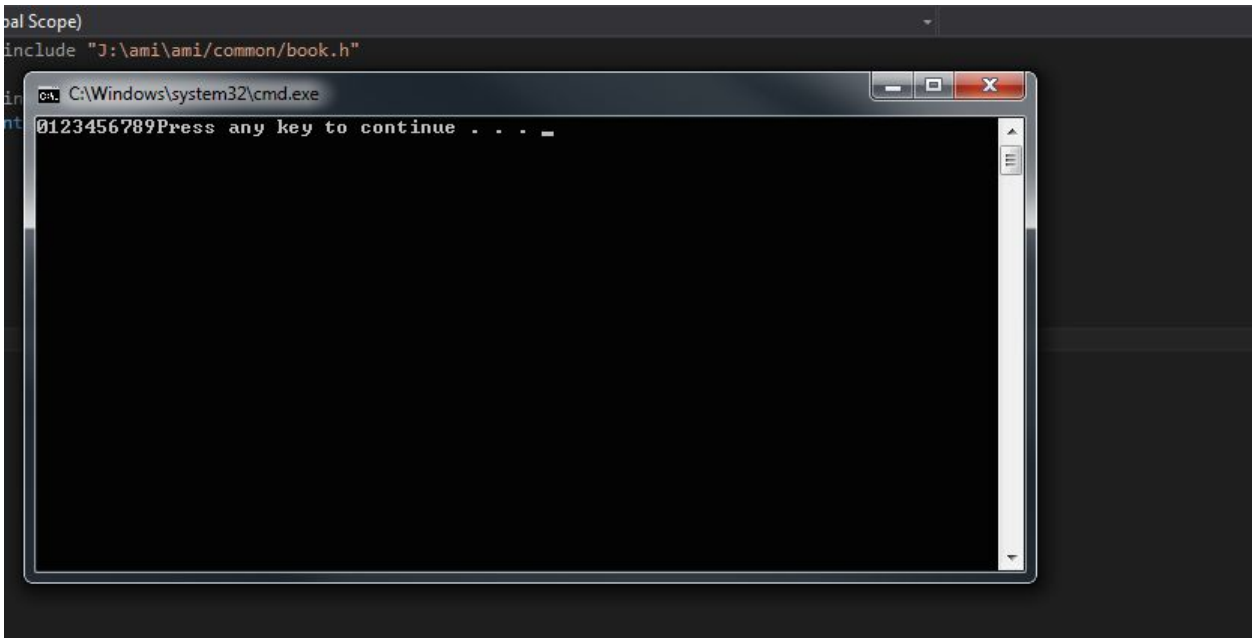
```
#include "J:\ami\ami/common/book.h"

int main() {
	int n[100],m[100];
	int i;
	for (i = 0; i < 100; i++) {
		n[i] = i+1;
		m[i] = i + 1;
	}
	m[44] = 34;
	n[44] = 34;
	for (int j = 0; j < 100; j++) {
		printf("%d\n", n[j]);
		printf("%d\n", m[j]);
	}
}
```

Answer: 2

```
#include "J:\ami\ami/common/book.h"
int main() {
	float n[10], m[10];
	int i=0;
	for (i > 0; i < 10; i++) {
		n[i] = i + 1;
		m[i] = i + 1;
	}
	n[4] = 12.34;
	m[4] = 12.34;
	for (int j = 0; j < 10; j++){
		printf("%f\n", n[j]);
	}
	for (int k = 0; k < 10; k++){
		printf("%f\n", m[k]);
	}
}
```

Answer: 3



Answer: 4

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

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

or we can change it to

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

Answer: 5

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

Is the correct way to call the kernel function.

Answer: 6

Changes to our kernel when N > 8,388,480 (65,535 x 128) :

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