Q1. You will implement the function $f(n) = n(n+a)(n-b)$ through multi-threading. The main thread will take three numbers n, a and b as input.

There will be 4 additional threads. Main thread will pass the number n as parameter to the first thread and the second threads. The numbers a and b will not be passed as parameter to the first and the second threads.

The first thread will calculate $n(n+a)$ and print it (according to given sample).

The second thread will use the $n(n+a)$ value calculated by the first thread and calculate $n(n+a)(n-b)$ and print the result (according to given sample).

The third thread will print "Done" after the above mentioned outputs.

The fourth thread will print necessary line breaks between outputs of other threads.

The order of the output lines should be as per following sample. Implement your proposed solution in a file named Q1.c.

Sample I/O	
Input	**Output**
5 2 3	Result1 = 35 Result2 = 70 Done
Input	**Output**
3 4 8	Result1 = 21 Result2 = -105 Done
Input	**Output**
-2 -3 -4	Result1 = 10 Result2 = 20 Done

Q2. Your program will print "This is Summer 2019". There will be 3 threads in addition to main thread. The 1st thread will print "This" and "Summer". The 2nd thread will print "is" and "2019". The third thread will print necessary spaces. Use semaphore(s)/mutex(s) for proper synchronization.

Implement your proposed solution in a file named Q2.c.

Q3. Design a program that has three additional threads.

1st thread takes input of two integers a and b.

2nd thread takes input of another integer c; values of c can be 1, 2, 3 or 4.

3rd thread performs calculation as per following rule.

c	Calculation
1	a + b
2	a - b
3	a * b
4	a / b

The main thread prints result of the calculation of 3rd thread.

Use semaphore(s)/mutex(s) for proper synchronization.

Implement your proposed solution in a file named Q3.c.

Q4. Consider the following program on the next page. Fix the program so that the program always produces the expected output (res = 2*NUM).

Implement your proposed solution in a file named Q4.c.

```
#include <pthread.h> // for pthread_create, pthread_join, pthread_exit
#include <stdio.h>   // for printf
#include <stdlib.h>  // for exit

#define NUM 1000000

int res = 0;

void* count(void *a)
{
    int i, tmp;
    for(i = 0; i < NUM; i++)
    {
        tmp = res;      /* copy the global res locally */
        tmp = tmp+1;    /* increment the local copy */
        res = tmp;      /* store the local value into the global res */
    }
}

int main(int argc, char * argv[])
{
    pthread_t tid1, tid2;

    if(pthread_create(&tid1, NULL, count, NULL))
    {
        printf("\n ERROR while creating thread 1");
        exit(1);
    }

    if(pthread_create(&tid2, NULL, count, NULL))
    {
        printf("\n ERROR while thread 2");
        exit(1);
    }

    if(pthread_join(tid1, NULL))       /* wait for the thread 1 to finish */
    {
        printf("\n ERROR while joining thread 1");
        exit(1);
    }

    if(pthread_join(tid2, NULL))       /* wait for the thread 2 to finish */
    {
        printf("\n ERROR while joining thread 2");
        exit(1);
    }

    if (res < 2 * NUM)
    {
        printf("\n BOOM! res is [%d], should be %d\n", res, 2*NUM);
    }
    else
    {
        printf("\n OK! res is [%d]\n", res);
    }

    pthread_exit(NULL);
}
```