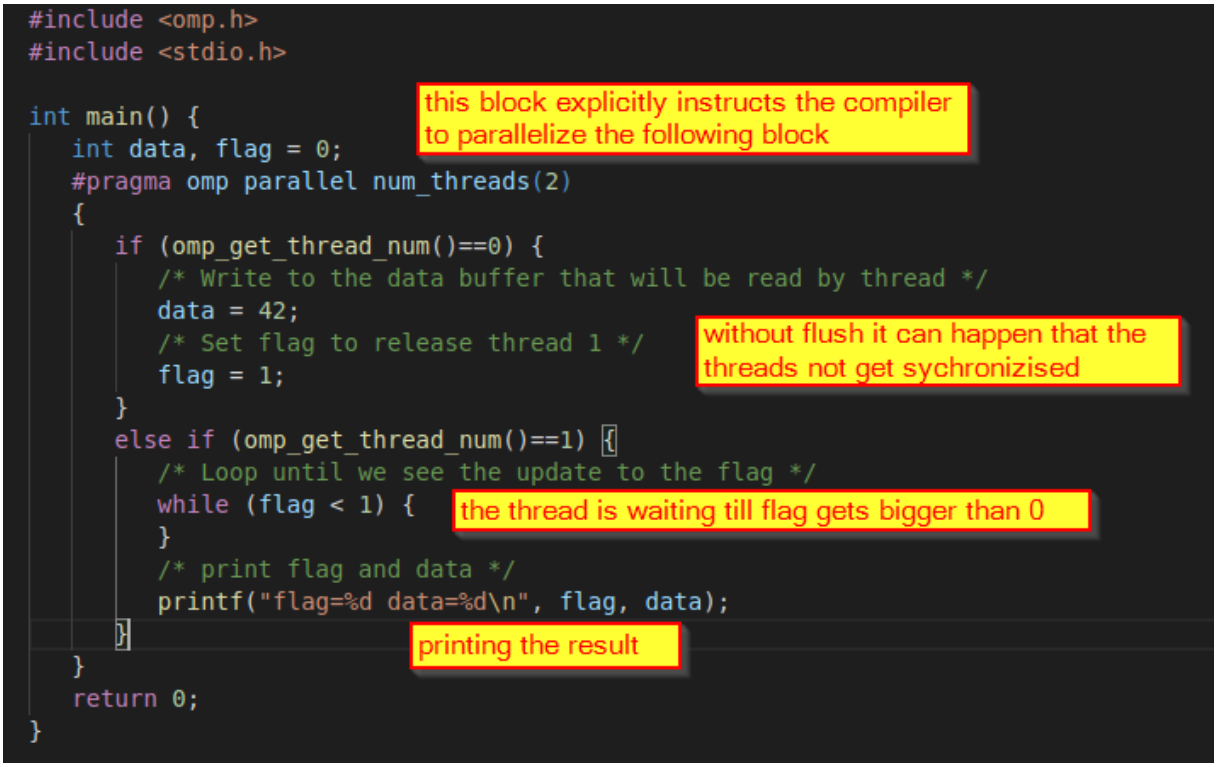
Exercise 2 **flush.c**

Team: Summit

What happens in the program?




```c
#include <omp.h>
#include <stdio.h>

int main() {
   int data, flag = 0;
   #pragma omp parallel num_threads(2)
   {
      if (omp_get_thread_num()==0) {
         /* Write to the data buffer that will be read by thread */
         data = 42;
         /* Set flag to release thread 1 */
         flag = 1;
      }
      else if (omp_get_thread_num()==1) {
         /* Loop until we see the update to the flag */
         while (flag < 1) {
         }
         /* print flag and data */
         printf("flag=%d data=%d\n", flag, data);
      }
   }
   return 0;
}
```

What observation can be made after running it multiple times?

The program sometimes works well and everything works like thought but there are cases in which the program not finishes because it stocks in an endless loop. This is because the threads not get synchronized safely and the flag never arrives to the second thread and so the program stays in the while loop.

Does this code require any "#pragma omp flush" directives?

Yes, because without we cannot be sure the threads get synchronized.

Solution:

```c
#include <omp.h>
#include <stdio.h>

int main() {
   int data, flag = 0;
   #pragma omp parallel num_threads(2)
   {
      if (omp_get_thread_num()==0) {
         /* Write to the data buffer that will be read by thread */
         data = 42;
         /* Flush data to thread 1 and relative to the flag */
         #pragma omp flush(flag, data)
         /* Set flag to release thread 1 */
         flag = 1;
         /* Flush flag to be sure that thread 1 sees the change */
         #pragma omp flush(flag)
      }
      else if (omp_get_thread_num()==1) {
         /* Loop until the update to the flag */
         #pragma omp flush(flag, data)
         while (flag < 1) {
            #pragma omp flush(flag, data)
         }

         #pragma omp flush(flag, data)
         /* print flag and data*/
         printf("flag=%d data=%d\n", flag, data);
      }
   }
   return 0;
}
```