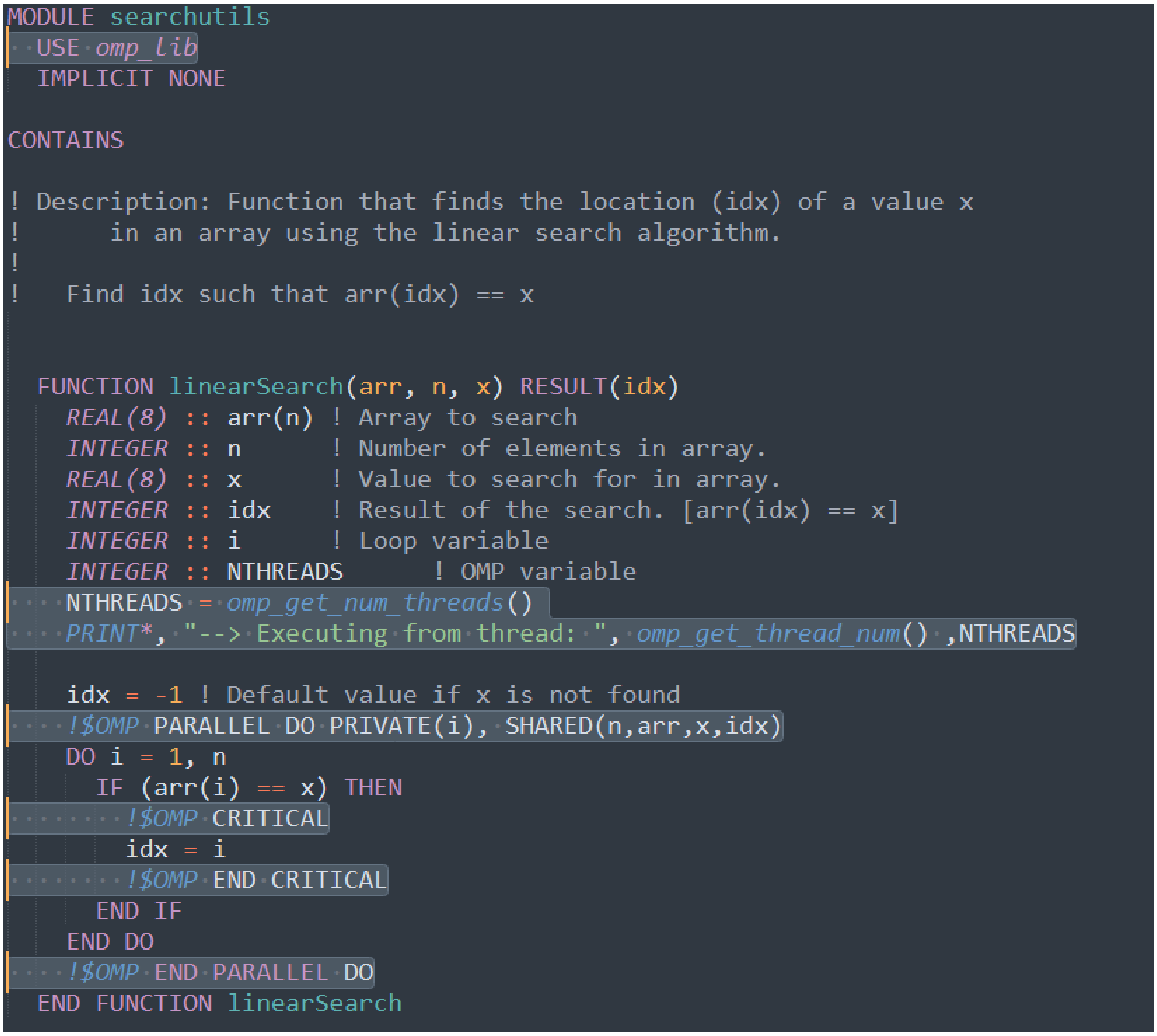
Homework 10

Juan Camilo Velasquez and Joaquin Rodriguez

Assignment 10: Parallelization of Fortran implementation of linear search algorithm. In this case Open MP is used to parallelize the for loop required during the linear search operation. The changes to the code are shown in Figure 1. For the implementation, the loop counter (i) was kept as private because each worker has its own inner counter, and the other variables that go into the operation are shared.



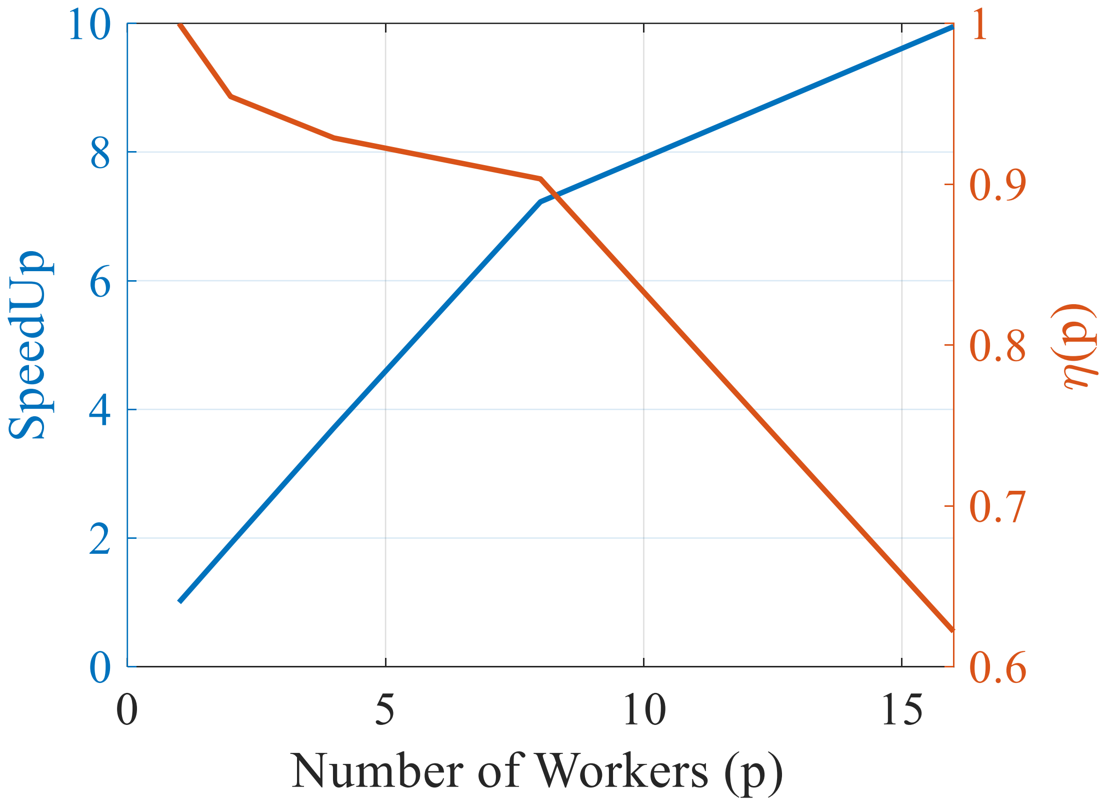
```
MODULE searchutils
  USE omp_lib
  IMPLICIT NONE

CONTAINS

! Description: Function that finds the location (idx) of a value x
!      in an array using the linear search algorithm.
!
!   Find idx such that arr(idx) == x


  FUNCTION linearSearch(arr, n, x) RESULT(idx)
    REAL(8) :: arr(n) ! Array to search
    INTEGER :: n      ! Number of elements in array.
    REAL(8) :: x      ! Value to search for in array.
    INTEGER :: idx    ! Result of the search. [arr(idx) == x]
    INTEGER :: i      ! Loop variable
    INTEGER :: NTHREADS       ! OMP variable
    NTHREADS = omp_get_num_threads()
    PRINT*, "--> Executing from thread: ", omp_get_thread_num() ,NTHREADS

    idx = -1 ! Default value if x is not found
    !$OMP PARALLEL DO PRIVATE(i), SHARED(n,arr,x,idx)
    DO i = 1, n
      IF (arr(i) == x) THEN
        !$OMP CRITICAL
        idx = i
        !$OMP END CRITICAL
      END IF
    END DO
    !$OMP END PARALLEL DO
  END FUNCTION linearSearch
```

Figure 1 Code modifications (highlighted)

The runtimes for the algorithm under 1,2,4,8 and 16 threads are shown in the following table:

OMP_NUM_THREADS	CPU Time (s)	Speedup	Efficiency
1	0.0394859	1.0000000	1.0000000
2	0.0206840	1.9090081	0.9545041
4	0.0106289	3.7149458	0.9287365
8	0.0054641	7.2264596	0.9033074
16	0.0039680	9.9510905	0.6219432

Also, the following figure contains the speedup and efficiency analysis of the code.



Conclusions: As seen in the results, this particular algorithm is highly parallelizable, and the speed up reaches up to ten for 16 workers, which is high compared to other codes that we have analyzed in the past. However, when compared with the runtimes from the previous assignment, with parallelization the linear search algorithm has still a very poor performance compared to the binary search.