```
/*  Example to illustrate  OMP reduction clause   */

#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

#define MAX_NUMS 2048
#define FNAME "nums"

int main()
{
/* Shared variables. */
   int * A ;
   int n   ;
   int total ;

/* Private variables. */
   int j, r, x, id, nprocs ;
   FILE * fp  ;

#pragma omp parallel shared(n,A,total) private(j,r,x,id,nprocs,fp)
   {
      nprocs = omp_get_num_threads() ;
      id = omp_get_thread_num()     ;

#pragma omp master
      {     /* The master thread does this. */
          A = (int *) malloc( MAX_NUMS * sizeof(int) ) ;
          if ( A == NULL ) {
              fprintf(stderr,"malloc failed !!\n" ) ;   exit(1) ;
          }
          fp = fopen( FNAME, "r" ) ;
          if ( fp == NULL ) {
             fprintf(stderr,"unable to read file '%s'\n", FNAME ) ;  exit(2) ;
          }
          n = 0 ;
          do {
             r = fscanf( fp, "%d", &x ) ;
             if ( ( r != EOF ) && ( n < MAX_NUMS ) ){
                A[n] = x ;
                n++ ;
             }
          } while ( r != EOF ) ;
          fclose( fp ) ;
          printf("%d numbers found.\n", n ) ;
          printf("%d procs.\n", nprocs ) ;

          total = 0 ;
      }  /* end section: allocate memory and read data.      */

/* Every process waits until the master thread has updated A, n, and total. */
#pragma omp barrier

/* Every thread participates in an omp for loop, with a reduction clause. */
#pragma omp for reduction(+:total)
   for ( j = 0 ; j < n ; j++ ) {
       total += A[j] ;
   }
#pragma omp master
{
        printf("Total = %d\n", total ) ;
}

}  /* End parallel section. */
} /* End main() */
```

```
------------------------- Sample Session -------------------------
cosmos% make
gcc -c -fopenmp -O3 reduce_demo.c
gcc -o reduce_demo -fopenmp -O3 reduce_demo.o
cosmos% ./reduce_demo
1024 numbers found.
11 procs.
Total = 519632
```