```c
/*  C  */

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

#define N 100000000


int main () {

   int* a;
   a = (int *) malloc(N*sizeof(int));
   if (a == NULL) {
      printf("Bad malloc of a, exiting\n");
      exit(-1);
   }

   for (int i = 0; i < N; i++) {
      a[i] = 1;
   }

// PART a
   double t = omp_get_wtime( );
   int s = 0;
   for (int i = 0; i < N; i++) {
      s = s + a[i];
   }
   t = omp_get_wtime( ) - t;
   printf("sequential reduction time is %f, value is %d\n", t, s);
   printf("\n\n\n\n");

// PART b
   int nt;
   #pragma omp parallel
   {
      #pragma omp single
      {
         nt = omp_get_num_threads( );
      }
   }

   int results[nt];
   for (int i = 0; i < nt; i++) {
      results[i] = 0;
   }

   t = omp_get_wtime( );
   #pragma omp parallel for
```

```
   for (int i = 0; i < N; i++) {
      results[omp_get_thread_num( )] += a[i];
   }

   s = 0;
   for (int i = 0; i < nt; i++) {
      s += results[i];
   }
   t = omp_get_wtime( ) - t;
   printf("hand-rolled parallel reduction time is %f, value is %d\n",
t, s);
   printf("\n\n\n\n");

// PART c
   s = 0;
   t = omp_get_wtime( );
   #pragma omp parallel for reduction(+: s)
   for (int i = 0; i < N; i++) {
      s += a[i];
   }
   t = omp_get_wtime( ) - t;
   printf("OMP reduction time is %f, value is %d\n", t, s);
   printf("\n\n\n\n");
}
```