OpenMP Programming Exercise Solution

```
/******************************************************************
* FILE: omp_A2.c
* DESCRIPTION:
* The program will assign a random value from the specified range to
* each array element in the parallel section of the code.
* On exit of the parallel section, the array will be sequentially
* checked to count the number of elements with the repeated value.
******************************************************************/
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>
/* Define the number of elements to be processed per thread*/
#define CHUNKSIZE  2
/* Define the size of the array that will store random numbers*/
#define N  10
/* Define the number of threads*/
#define NUM_THREADS 4
/* define the random number range as follows*/
#define MAXRND 25

int main (int argc, char *argv[])
{
int nthreads, tid, i, chunk;
int min, max, count, j;
int a[N];
int repeated[N+MAXRND] = { 0 };

/* Initialise */
chunk = CHUNKSIZE;
omp_set_num_threads(NUM_THREADS);

#pragma omp parallel shared(a,nthreads,chunk) private(i,tid,min,max)
  {
  tid = omp_get_thread_num();
  if (tid == 0)
    {
    nthreads = omp_get_num_threads();
    printf("Number of threads = %d\n", nthreads);
    }
  printf("Thread %d starting...\n",tid);
  srandom(time(NULL) | tid);
  min=1; max=MAXRND;

  #pragma omp for schedule(dynamic,chunk)
  for (i=0; i<N; i++)
    a[i] = (random()%(max-min+1))+min;

 } /* end of parallel section */

/* print the computed random valued matrix sequentially*/
for (i=0; i < N; i++){
 printf(" a[%d]:%d\n",i, a[i]);
 }
```

OpenMP Programming Exercise Solution

```
/*count the repeat occurrences of a value in a[]*/
printf("-----------------------------------\n");
printf("a[]:No_of_Copies_Found\n");
for ( i = 0 ; i < N ; i++ ){
     printf("%d -> ", repeated[a[i]]);
     repeated[a[i]]++;
     printf("a[%d]:%d -> %d\n",i,a[i],repeated[a[i]]);
   }
   count=0;
/* repeated[] = 1 represents unique occurrences of random numbers -
ignored*/
for ( i = 0 ; i < N+MAXRND; i++ )
   if (repeated[i] > 1)
   count++;
printf("Total:(num of two or more matching vals found in the a[]) ->
%d\n", count);
return 0;
}
```

Output

```
Thread 1 starting...
Number of threads = 4
Thread 2 starting...
Thread 3 starting...
Thread 0 starting...
 a[0]:20
 a[1]:21
 a[2]:22
 a[3]:17
 a[4]:1
 a[5]:21
 a[6]:11
 a[7]:2
 a[8]:23
 a[9]:2
-----------------------------------
a[]:No_of_Copies_Found
0 -> a[0]:20 -> 1
0 -> a[1]:21 -> 1
0 -> a[2]:22 -> 1
0 -> a[3]:17 -> 1
0 -> a[4]:1 -> 1
1 -> a[5]:21 -> 2
0 -> a[6]:11 -> 1
0 -> a[7]:2 -> 1
0 -> a[8]:23 -> 1
1 -> a[9]:2 -> 2
Total:(num of two or more matching vals found in the a[]) -> 2
real    0m0.008s
user    0m0.004s
sys     0m0.000s
```

Compile & Run

user@ubuntu: gcc omp_A2.c –fopenmp
user@ubuntu: time ./a.out