PARALLEL AND DISTRIBUTED COMPUTING
L – 19,20
DATE: 29.8.19
FACULTY : PROF. GAYATHRI R.
DHRUBANKA DUTTA, 17BCE1019

Exercise :
Implement producer consumer problem using openmp.

CODE :

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


#define MAXWORK 40

int work[MAXWORK],  // work to be done
    nwork=0,  // number of items in the queue
    nextput=0,  // producer will place number # at work[nextput]
    nextget=-1,  // consumer will obtain next # at work[nextget]
    breaksum,  // sum after which everyone stops
    done = 0,  // value 1 signals producer exceeded breaksum
    psum,csum,  // sums found by producer, consumers
    pwork,  // work done by producer
    *cwork,  // work done by the consumers
    nth,  // number of threads
    debugflag;  // 1 if debug

void next(int *m)
{  (*m)++;
   if (*m >= MAXWORK) *m = 0;
}

void putwork(int k)
{  int put = 0;
   while (!put)  {
      if (nwork < MAXWORK)  {
         #pragma omp critical
         {  work[nextput] = k;
            if (nwork == 0) nextget = nextput;
            next(&nextput);
            nwork++;
            put = 1;
         }
      }
```

```
      else sched_yield();
   }
}

int getwork()
{  int k,get=0;
   while (!get)  {
      if (done && nwork == 0) return -1;
      if (nwork > 0)  {
         #pragma omp critical
         {
            if (nwork > 0)  {
               k = work[nextget];
               next(&nextget);
               nwork--;
               if (nwork == 0) nextget = -1;
               get = 1;
            }
         }
      }
      else sched_yield();
   }
   return k;
}

void dowork()
{
   #pragma omp parallel
   {  int me = omp_get_thread_num(),
          num;
      #pragma omp single
      {  int i;
         nth = omp_get_num_threads();
         printf("there are %d threads\n",nth);
         cwork = (int *) malloc(nth*sizeof(int));
         for (i = 1; i < nth; i++) cwork[i]=0;
      }
      if (me == 0 && debugflag) {int wait=0; while (!wait) ; }
      #pragma omp barrier
      if (me == 0)  {  // I'm the producer
         pwork = 0;
         while (1)  {
            num = rand() % 100;
            putwork(num);
            psum += num;
            pwork++;
            if (psum > breaksum)  {
               done = 1;
               return;
```

```
        }
      }
    }
    else  {  // I'm a consumer
      while (1)  {
        num = getwork();
        if (num == -1) return;
        cwork[me]++;
        #pragma omp atomic
        csum += num;
      }
    }
  }
}

int main(int argc, char **argv)
{  int i;
  breaksum = atoi(argv[1]);
  debugflag = atoi(argv[2]);
  dowork();
  printf("sum reported by producer:  %d\n",psum);
  printf("sum reported by consumers:  %d\n",csum);
  printf("work done by producer:  %d\n",pwork);
  printf("work done by consumers:\n");
  for (i = 1; i < nth; i++)
    printf("%d\n",cwork[i]);
}
```

OUTPUT :