12. Design develop and run a multi-threaded program to generate and print Fibonacci series. One thread has to generate the numbers up to the specified limit and Another thread has to print them. Ensure proper synchronization

```
# include<stdio.h>
# include<omp.h>
# include<stdlib.h>
int MAX;
      int Fibonacci(int n)
      {
            int x, y;
            if (n < 2)
                  return n;
            else
            {
                  x = Fibonacci(n - 1);
                  y = Fibonacci(n - 2);
                  return (x + y);
            }
      }

      int FibonacciTask(int n)
      {
      int x, y;
      if (n < 2)
            return n;
      else
            {
                  x = Fibonacci(n - 1);
                  y = Fibonacci(n - 2);
                  return (x + y);
      }
      }

      int random_num()
      {
          int temp;
            temp = rand();
            temp = temp%24;
            MAX = temp;
            return(MAX);
      }

      int main(int argc, char * argv[])
      {

            int FibNumber[25] = {0};
      int j, temp,tmp,id,i = 0;
            int n, tid, nthreads;

            printf("Please Enter the number Range :");
```

```
            scanf("%d",&n);
            printf("\n");
            omp_set_num_threads(2);

            //Parallel region
            #pragma omp parallel
            {
                  printf("The number of threads are
%d\n",omp_get_num_threads());
            #pragma omp for private (tid, tmp, FibNumber)
            for(j = 1;  j<=n; j++)
            {
                  tmp = random_num();
                  /* Get thread number */
                  /*    tid = omp_get_thread_num();
                        printf("The number of threads are
%d\n",omp_get_num_threads());
                        printf("The thread id  is = %d\n", tid); */
                  /* The critical section here will enable, not more
then one
                     thread to execute in this section (synchronization)
*/
                  #pragma omp critical
                        {
                              /* Get thread number */
                        /* tid = omp_get_thread_num();
printf("********************* inside critical section
*****************\n");
                        printf("The thread id is  = %d\n", tid); */
                        for(i = 1; i <= tmp; i++)
                                    FibNumber[i] = FibonacciTask(i);
                     printf("The number value is %d:",tmp);
                        for(i = 1; i <= tmp; i++)
                              printf("%d \t", FibNumber[i]);
                              printf("\n\n");
                        }
                  }
            }

}
```

Command For Execution

cc -fopenmp 12.c

./a.out