Another test of pseudo random numbers is to compute their “moments”. These are defined by

\[ m_1 = \frac{1}{N} (r_1 + r_2 + r_3 + r_4 + \cdots + r_N) \]
\[ m_2 = \frac{1}{N} (r_1^2 + r_2^2 + r_3^2 + r_4^2 + \cdots + r_N^2) \]
\[ m_3 = \frac{1}{N} (r_1^3 + r_2^3 + r_3^3 + r_4^3 + \cdots + r_N^3) \]
\[ m_4 = \frac{1}{N} (r_1^4 + r_2^4 + r_3^4 + r_4^4 + \cdots + r_N^4) \]

In other words, to get the \( p \)th moment you raise each number to power \( p \) and then average.

On page two is a program to compute the first six moments. Type it in and run it. Use 1000000 random numbers. What do you get for the moments? Do you see any pattern?

**Comments:**

[1] Can you prove the pattern? (This is really a calculus problem.)

[2] At the beginning of the week we discussed computer speeds and suggested a computer could do about \( 10^9 \) operations per second. Is this estimate consistent with the performance of your code?

[3] Actually, you can define the moments of any distribution of numbers, not just our random numbers which are uniform on [0, 1]. We will not go into that here.
```c
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

int main()
{
    srand(time(NULL));
    int i,N;
    double R,sum1=0.,sum2=0.,sum3=0.,sum4=0.,sum5=0.,sum6=0.;
    printf("Enter the number of random numbers used ");
    scanf("%d",&N);
    for(i=0;i<N;i++)
    {
        R=(double)rand()/RAND_MAX;
        sum1=sum1+R;
        sum1=sum1+R;
        sum2=sum2+R*R;
        sum3=sum3+R*R*R;
        sum4=sum4+R*R*R*R;
        sum5=sum5+R*R*R*R*R;
        sum6=sum6+R*R*R*R*R*R;
    }
    printf("%lf\n",sum1/N);
    printf("%lf\n",sum2/N);
    printf("%lf\n",sum3/N);
    printf("%lf\n",sum4/N);
    printf("%lf\n",sum5/N);
    printf("%lf\n",sum6/N);
}
```