Solution Lab 07

Task 1: Write a program (using a for loop) that generates pair of 10 random integers. In pair first integer (say x) should be in range 2-9 and second integer (say y) in the range 2-15 and find X^Y. See test run for further understanding and output format:

```
3^7 = 2187
9^3 = 729
3^15 = 14348907
...
```

```
      int i, n1, n2;
      for (i=1;i<=10;i++){
            n1 = rand() % 8 + 2;
            n2 = rand() % 14 + 2;
            printf ("%d^%d = %1.0f\n", n1, n2, pow(n1, n2));
      }
```

Task 2: Write a program that input an integer (positive, don't check just give message) and check whether number is perfect square or not. See test run for further understanding and output format:

Hint: Run loop (say 1 to n/2) and check equality of n with square (a*a) of loop variable, if equal it means perfect square.

```
Enter a positive non-zero integer: 25
25 is perfect square
Enter a positive non-zero integer: 22
22 is not perfect square
```

```
      int i, n;
      printf ("Enter a positive non-zero integer: ");
      scanf ("%d", &n);
      if (n <= 0 ){
            printf ("Sorry, wrong input\n");
            return 0;
      }
      for (i=1;i*i<n;i++);
      if (i*i == n)
            printf ("%d is perfect square\n", n);
      else
            printf ("%d is not perfect square\n", n);
```

Task 3: Write a program that input 10 integers at random using loop, print them. At the end give count, how many of them are odd and how many are even. See test run for further understanding and output format:

```
23904 29616 17914 9180 3749 30962 21058 29981 23004 30365
Count of even numbers: 7
Count of odd numbers: 3
```

```
    int i, n, countE = 0;
    for (i = 1 ; i <= 10 ; i++ ){
        n = rand() ;
        printf ("%d ", n);
        if ( n % 2 == 0 )     countE++;
    }
    printf ("\nCount of even numbers: %d", countE);
    printf ("\nCount of odd numbers: %d", 10 - countE);
```

Task 4: Write a program that input an integer (positive, don't check just give message) and find square root (do not use built-in function) by following method:

1. Say number (input value) is n
2. Guess some random value (say s1) less than number n
3. Find s2 using formula: $s2 = \frac{s1+\frac{n}{s1}}{2}$
4. if $abs\ (s2 - s1\ ) < 0.00001$ s1 is your answer, otherwise
5. Assign value of s2 to s1 and repeat step 3

A working example is shown in the box for integer 29 and same result is generated by my program, see both working as well as sample run for format, as well as for understanding:

n	**29**
s1	25
s2	13.0800
Diff	11.9200
s1	13.0800
s2	7.6486
Diff	5.4314
s1	7.6486
s2	5.7201
Diff	1.9285
s1	5.7201
s2	5.3950
Diff	0.3251
s1	5.3950
s2	5.3852
Diff	0.0098
s1	5.3852
s2	5.3852
Diff	0.0000

```
Enter a positive non-zero integer: 29
Square Root of 29 is 5.39
Enter a positive non-zero integer: 36
Square Root of 36 is 6.00
```

```
    int n;
    float s0, s1, d;
    printf ("Enter a positive non-zero integer: ");
    scanf ("%d", &n);
    if (n <= 0 ){
        printf ("Sorry, wrong input\n");
        return 0;
    }
    s1 = rand() % n;
    do{
        s0 = s1;
        s1 = ( s0 + n / s0 ) / 2 ;
        d = s0 - s1;
        if ( d < 0 )    d = -d;
}while ( d > EPSILON);
printf ("Square Root of %d is %.2f\n", n, s0);
```

[END OF LAB :) :) :)]