Prog. and Data Structures Lab (Section -1) Spring 2021

Assignment Set 2

---

1. **Series Summation.** Computers often use Taylor series expansions for computing the values of mathematical functions. For example, consider the following Taylor series expansion:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} \ldots$$

Let us consider the n^{th} term: $\frac{x^n}{n!}$ in this series. It is not a good idea to compute x^n and $n!$ separately because both of these can be VERY large for large values of n (and computers can store numbers of a limited size). Instead, we can compute the n^{th} term in terms of the $(n-1)^{th}$ term as follows:

$$t_n = \frac{x^n}{n!} = \left(\frac{x}{n}\right)\frac{x^{n-1}}{(n-1)!} = \left(\frac{x}{n}\right) t_{n-1}$$

Write a C program which uses the same approach for finding the values of $sin\,(x)$ and $cos\,(x)$. Your program must read the value of x and k from the user. Then it must compute the values of these functions and print the values computed using this method. For comparison, you must also print the values returned by the math library functions `sin()` and `cos()` for the same value of x.

The Taylor series expansions are given below:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \ldots$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} \ldots$$

2. **Computing Mean and Standard Deviation without storing the numbers.** The formulae for computing mean, μ, and variance, σ^2, are as follows:

$$\mu = \frac{\sum_{i=1}^{N} x_i}{N}$$

$$\sigma^2 = \frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}$$

Study the following code snippet for computing the sum of a set of numbers without storing them:

```
sum = 0; count = 0;
c = 'Y';
while ((c == 'y')||(c == 'Y')) {
      printf("Enter the number: ");
```

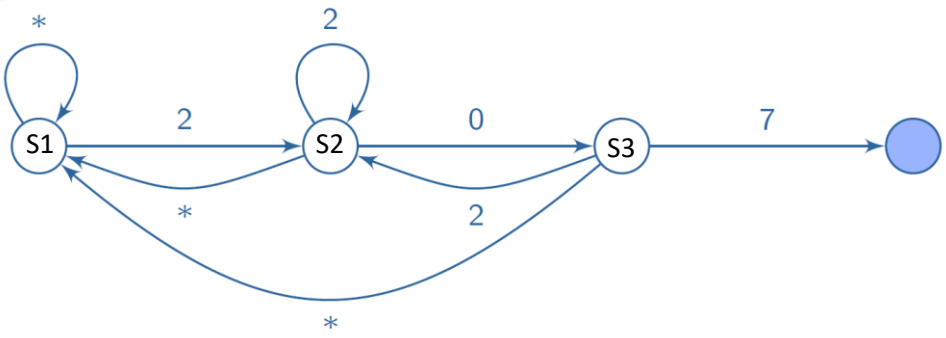
```
        scanf("%f", &x);
        sum = sum + x;
        count++;
        printf("Any more numbers? (Y/N) ");
        c = getchar();
}
printf("The sum is %f", sum);
```

Write a C program by modifying this code, so that the program computes and prints the mean, μ, the variance, σ^2, and the standard deviation, σ of the numbers. Note that the input numbers should be read only once, and not more than one of the input numbers should be stored at any time.

Hint:

$$\sigma^2 = \frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N} = \frac{\sum_{i=1}^{N}(x_i^2 - 2x_i\mu + \mu^2)}{N} = \frac{\sum_{i=1}^{N} {x_i}^2}{N} - 2\mu\frac{\sum_{i=1}^{N} x_i}{N} + \mu^2$$

3. **Numerical Door Lock.** A door lock opens when the code **207** is pressed in its keypad. The user may press any sequence of keys, numbered 0, 1, …, 9, but the lock does not open until the user presses 2 followed by 0 followed by 7. Since this is a 3-digit code, the lock needs to remember the previous two key presses to determine whether 207 has been pressed in sequence. In programming we often need to model such *states* of the device we wish to program. For example, the figure shown below is a state machine for the numerical door lock with code 207.



The rightmost (blue) state is reached when the previous three key presses are 2-0-7 and the lock is open. State S3 represents the situation where the previous two key presses are 2-0 (and therefore the first two digits of the key has matched). If the next key press is 7, then the lock moves to the blue state and opens. State S2 represents the situation where the previous key press was 2 (and therefore the first digit of the key has matched). From this state it goes to S3 if the next key pressed is 0. S1 represents the state where no part of the key has matched. The labels on the transitions (edges) between the states are the key presses. We use * to denote a wildcard or default transition. From a state we take the default transition when the next key press does not match with any of the other transitions from that state.

Your task is to write a C program that continues to read a sequence of digits, one at a time, until the digits 2, 0, 7 arrive one after the other. When that happens, the program must print: `Lock is open` and terminate. The following code skeleton is provided as a hint:

```
state = 1;   /* Initial state is S1 */
open = 0;    /* This is the status of the lock. open==0 means lock is closed */
while (open == 0) {
      c = getchar(); /* Read a digit from the keypad */
      switch (state) {
            case 1: /* You need to complete this part */ break;
            case 2: /* You need to complete this part */ break;
            case 3: if (c == '2') state = 2;
                        else if (c == '7') {
                              open = 1;
                              printf("Lock is open \n");
                        }
                        else state = 1;
                        break;
      }
}
```

[Point to ponder: How would things change is the code was 22022 ? This is not part of the assignment, but good to think about.]