CSE 3100 Systems Programming
Homework #1

Note

This first problem set contains four problems. Those are about numerical computations. See the PDF below for details. Let me emphasize that you should be *extremely* careful about the formatting of your input. Remember that the automatic grading will expect a specific formatting, so you should not deviate from that.

1 Computing e (33 points)

In this exercise, we use the following fact to approximate e:

$$e = \sum_{k=0}^{\infty} \frac{1}{k!} \tag{1}$$

To approximate e, we modify the above summation so that k is summed from 0 to n, where n is a non-negative integer.

$$e \approx \sum_{k=0}^{n} \frac{1}{k!} \tag{2}$$

We provide code that reads n from the user. You need to fill in the rest. Below are the outputs from a few example runs. These can be used to check the correctness of your code:

```
$ ./e
n = 3
e =  2.666667
$ ./e
n = 4
e =  2.708333
```

2 Modular Exponentiation (33 points).

The following recurrence specifies modular exponentiation

$$n^e \mod m$$

where n, e, and m are integers such that $n > 0$, $e > 0$, and $m > 1$. Namely:

$$n^e \mod m = \begin{cases} \left(n^2 \mod m\right)^{\frac{e}{2}} \mod m & \Leftrightarrow \quad e \text{ is even} \\ \left(\left(n^2 \mod m\right)^{\frac{e}{2}} \mod m\right) \cdot n \mod m & \Leftrightarrow \quad e \text{ is odd} \end{cases}$$

Write a C program `powerMod.c` that prompts the user for three integers n, e, and m and then prints to the standard output the integer $n^e \mod m$. You can assume that $n > 0$, $e > 0$, and $m > 1$ and all integers are less than $2^{31} - 1$. You do not have to cope with incorrect inputs. However, your implementation should be *iterative* since we have not yet discussed functions.

The block below shows the expected execution of the program at the command line:

```
$ ./powerMod
Please enter n, e, and m: 123 456 789
123 ** 456 mod 789 = 699
```

Specifically, the powerMod executable first prints the string "Please enter n, e, and m:" to prompt the user to enter the numbers n, e, and m. The user types three integers (123, 456, and 789 in this example), separated by whitespaces. When the user hits the return key the program computes the desired value and produces the output "123 ** 456 mod 789 = 699" on a single line.

3 Running Min and Max(34 points)

Write a C program `extremes.c` that reads floating-point numbers (double) from the standard input and, after reading each number, prints the minimum and maximum numbers that has been read so far . The program must terminate when there is an error or the end of file (EOF) is detected at the standard input (e.g., when the user presses Ctrl-D).

The block below shows a sample execution of the program in which the user provided input consists, in this order, of numbers 1, 2, 1e10 (10^{10}), and 0:

```
$  ./extremes
1
Min=1.000000 Max=1.000000
2
Min=1.000000 Max=2.000000
1e10
Min=1.000000 Max=10000000000.000000
0
Min=0.000000 Max=10000000000.000000
```

Notes. The loop needed to read the input is as follows:

```
while (scanf("%lf", &x) == 1) {  // pay attention to %lf
    ...
};
```

Here `x` is a `double` variable used to store the number read in each iteration. Printing the running min and max formatted as in the example above can be done using

```
printf("Min=%lf Max=%lf\n", min, max);
```

where `min` and `max` are variables of type `double`.