Name: _____

Date: _____

Question:	1	2	Total
Points:	20	65	85
Score:			

Representation of floating point numbers

1. You are developing a new standard for floating point arithmetic for microchips. It proposes to store floating point numbers in 14 bits, in a manner similar to IEEE754 standard: one bit for the sign, five bits for the exponent, and (one plus) eight bits for the fractional part of the number. You are **not** reserving special values of the exponent for zero, infinity, and NaN.

   (a) (5 points) What is the smallest positive floating point number in your system?

   _____

   (b) (5 points) What is the largest floating point number?

   _____

   (c) (5 points) Approximately, how many floating point numbers are in your system?

   _____

   (d) (5 points) What is machine ϵ in your system?

   _____

Programming

2. Earlier in the semester we consider the following integral:

$$I_n = \int_0^1 x^n e^{-x} \mathrm{d}x, \tag{1}$$

   where n is integer non-negative parameter, $n = 0, 1, 2, \ldots$

   Looking for an elegant and numerically efficient way to evaluate integrals Eq. (1), we derived the following recurrence relation for I_n:

$$I_n = n\, I_{n-1} - \frac{1}{e}, \tag{2}$$

and supplemented it by the "initial" condition:

$$I_0 = \int_0^1 e^{-x} \mathrm{d}x = 1 - \frac{1}{e}. \tag{3}$$

Unfortunately the program based on Eq. (2), (3) failed. It calculated I_n for small n correctly but produced meaningless results for $n \gtrsim 18$.

As discussed in class on Oct 29, we now understand the reason for a failure of the original program as well as the origin of the number 18 above.

Now, your task is to adapt the recurrence relation such that it can be used to produce accurate results for any n, write a function that calculates a set of integrals for $1 \leq n \leq 100$, and compare the performance of your algorithm with the performance of a good-quality general integration program.

(a) (5 points) rewrite Eq. (1) to act in the stable direction of the recursion, and supplement it by an initial condition

(b) (5 points) Write a function with the following declaration

```
1    void integral_recur(int nmin, int nmax, double vals[]);
```

that uses your new algorithm, calculates I_n for $\mathrm{nmin} \leq n \leq \mathrm{nmax}$, and stores the results into the user-provided array `vals`.

(c) (10 points) Write a function with the following declaration

```
1    void integral_gen (int nmin, int nmax, double vals[]);
```

that uses a high quality general integrator, calculates I_n for $\mathrm{nmin} \leq n \leq \mathrm{nmax}$, and stores the results into the user-provided array `vals`.

As a general integrator, the function `gsl_integration_qag` from GSL library is recommended. Use $\epsilon = 10^{-9}$ for both the absolute and the relative error.

In your main program allocate the storage for the array values as following:

```
1    #define NMAX 100
2
3    double vals1[NMAX + 1], vals2[NMAX + 1];
```

(d) (10 points) write a program that compares the results of `integral_recur` and `integral_gen`, thus validating your code.

(e) (15 points) Refactor your program by adding code that measures the time per function call of the two functions. To increase the precision of the time measurements repeat functions calls multiple times such that the total running times for each algorithm is between 1 and 2 seconds.

You are welcome to recycle the code your developed for Midterm I project.

Record time per function call for both algorithms, t_{recur} and t_{gen}, as well as the ratio $t_{\mathrm{gen}}/t_{\mathrm{recur}}$ (rounded to the nearest integer).

t_{gen}: _____ t_{recur}: _____ $t_{\mathrm{gen}}/t_{\mathrm{recur}}$: _____

(f) (10 points) Your C code should be well commented, written elegantly, and properly and consistently formatted.

(g) (10 points) Prepare README.md file and upload the standard set of files for your project to GitHub to the folder named **hw07**. Provide the link to the project:

https/github.com/ _____