Practice Compete Jobs Rank Leaderboard Buritomath

Dashboard > Functional Programming > Introduction > Evaluating e^x

Badge Progress (Details)

Points: 133.00 Rank: 5395

Evaluating e^x

by idlecool

Problem Submissions Leaderboard Discussions

The series expansion of e^x is given by:

$1 + x + x^2/2! + x^3/3! + x^4/4! + \ldots\ldots$

Evaluate e^x for given values of x by using the above expansion *for the first* 10 *terms*.

Input Format

The first line contains an integer N, the number of test cases.
N lines follow. Each line contains a value of x for which you need to output the value of e^x using the above series expansion. These input values have exactly 4 decimal places each.

Output Format

Output N lines, each containing the value of e^x, computed by your program.

Constraints

$1 <= N <= 50$
$-20.00 <= x <= 20.00$
Var, *Val* in Scala and *def* and *defn* in Clojure are blocked keywords. The challenge is to accomplish this without either mutable state or direct declaration of local variables.

Sample Input

```
4
20.0000
5.0000
0.5000
-0.5000
```

Sample Output

```
2423600.1887
143.6895
1.6487
0.6065
```

Explanation

The output has the computed values of e^x corresponding to each test case. They are correct up to 4 decimal places and on separate lines.

Scoring

All test cases carry an equal weight in the final score. For your solution to pass a given test case, all the values of e^x computed by you must be within $+/-0.1$ of the expected answers. This tolerance level has been kept to account for slightly different answers across different languages.

Solved score: 20.00pts

Submissions: 7507
Max Score: 20
Difficulty: Easy

Rate This Challenge:
☆☆☆☆☆

More

If you cannot find your favorite language here, please check out the algorithm domain. We support over 40 popular languages.

Current Buffer (saved locally, editable) Racket

```racket
#lang racket
; Enter your code here. Read input from STDIN. Print output to STDOUT

(define (factorial n) (if (= n 0) 1 (* n (factorial (- n 1)))))

(define (calculate-e-term term n)
  (cond ((= term 0) 1)
        ((= term 1) n)
        (else (/ (expt n term) (factorial term)))))

(define (calculate-e n) (foldr + 0 (map (lambda (x) (calculate-e-term x n)) (range 10))))

(define (read-numbers n)
  (cond ((= n 1) (displayln (~r (calculate-e (read)) #:precision 4)))
        (else
         (begin
           (displayln (~r (calculate-e (read)) #:precision 4))
           (read-numbers (- n 1))))))

(read-numbers (read))
```

Line: 1 Col: 1

Upload Code as File   Test against custom input   Run Code   Submit Code

Join us on IRC at #hackerrank on freenode for hugs or bugs.

Contest Calendar | Interview Prep | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy | Request a Feature