C200 Programming Assignment № 3

Dr. M.M. Dalkilic
Computer Science
School of Informatics, Computing, and Engineering

Indiana University, Bloomington, IN, USA

September 22, 2022

Introduction

Due Date: Thursday September 29, 2022, 11:00 PM EST

Student Pairs are provided at the end of this document.
Please read through the problems carefully. Some reminders from lecture:

- We will include the math module
- You are not allowed to use any functions outside of one's we've used. For this homework, you cannot use in, max, min.
- A **constant** function is a function whose output value is the same for every input value. For example: f(x) = 3, irrespective of what we plug in for x, the function f will allways output 3.
- Now we know that a **constant** (or fixed) function $f(x)$ returns the same constant value, *e.g.*,

$$f(x, y) = 3 \tag{1}$$

  No matter the inputs, the value is three.

- Some of you have asked about this. We can convert between $\log_a, \log_k$:

$$\log_b(x) = \frac{\log_k(x)}{\log_k(b)} \tag{2}$$

  Why? Remember that if $\log_b(x) = r$, then $b^r = x$. So, let's first write

$$\log_b(x) = r \tag{3}$$
$$\log_k(x) = s \tag{4}$$
$$\log_k(b) = t \tag{5}$$

This means

$$
\begin{aligned}
b^r &= x && (6)\\
k^s &= x && (7)\\
k^t &= b && (8)
\end{aligned}
$$

We see that $b^r = x = k^s$. Since $b = k^t$, we can write:

$$(k^t)^r = k^{rt} = x = k^s \quad (9)$$

Then

$$
\begin{aligned}
\log_k(k^{rt}) &= \log_k(x) = \log_k(k^s) && (10)\\
rt &= \log_k(x) = s && (11)
\end{aligned}
$$

Using equations 3,5 for r, t we have

$$
\begin{aligned}
\log_b(x)\log_k(b) &= \log_k(x) && (12)\\
\log_b(x) &= \frac{\log_k(x)}{\log_k(b)} && (13)
\end{aligned}
$$

Problem 1: Functions and math module

All the following functions are drawn from real-world sources. This is an exercise for you to learn how to use a module on your own. From the math module use

- `math.exp(x)` for e^x
- `math.ceil(x)` for $\lceil x \rceil$ rounds x UP to the nearest integer k such that $x \leq k$
- `math.log(x)` for $\log_e(x) = \ln(x)$.

1. According to the Center for Disease Control (CDC) the bacteria *Salmonella* causes about 20K hospitalizations and nearly 400 deaths a year. The formula for how fast this bacteria grows is:

$$N(n_0, m, t) = n_0 e^{mt} \quad (14)$$
$$(15)$$

   where n_0 is the initial number of bacteria, m is the growth rate *e.g.*, 100 per hr, and t is time in hours. Here is how you would calculate the size for an initial colony of 500 that grows at the rate of 100 per hr, for four hours.

$$N(500, 100, 4) = 2.610734844882072 \times 10^{176} \quad (16)$$

2. The number of teeth $N_t(t)$ after t days from incubation for *Alligator mississippiensis* is:

$$N_t(t) = 71.8e^{-8.96e^{-0.0685t}} \quad (17)$$
$$N_t(1000) = \lceil 71.8 \rceil = 72 \quad (18)$$

3. If we want to calculate the work done when an ideal gas expands isothermally (and reversibly) we use for initial and final pressure $P_i = 10\ bar, P_f = 1\ bar$ respectively at $300^o K$. In this problem we are using $\ln$ which is $\log_e$ ('math.log uses base e by default'). Because $\log_e$ is used so often, you'll see it just as often abbreivated as $\ln$.

$$W(P_i, P_f) = RT \ln(P_i/P_f) \quad (19)$$
$$W(10, 1) = \lceil 8.314(300)(\ln 10) \rceil = 5744 \quad (20)$$

   at temperature T (Kelvin) and $R = 8.314\ J/mol$ the universal gas constant.

4. The Wright Brothers are known for their Flyer and its maiden flight. The formula for lift is:

$$L(V, A, C_\ell) = kV^2 A C_\ell \quad (21)$$
$$L(33.8, 512, 0.515) = \lceil 0.0033(33.8)^2(512)0.515 \rceil = 995 \quad (22)$$

where k is Smeaton's Coefficient ($k = 0.0033$ from their wind tunnel), $V = 33.8\ mph$ is relative velocity over the wing, $A = 512\ ft^2$ area of wing, and $C_\ell = 0.515$ coefficient of lift. The Flyer weighed $600\, lbs$ and Orville was about $145\, lbs$. We can see that the lift is sufficient since $995 > 745$ (where 745 = combined weight of flyer and Orville).

Deliverables for Problem 1

- Complete the functions described above.

Problem 2: Quadratic

We saw, and also know from basic algebra that for $ax^2 + bx + c = 0$ the roots (values that make the equation zero) are given by:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{23}$$

The expression $b^2 - 4ac$ is called the discriminant. By looking at the discriminant we can deteremine properties of the roots:

- If $b^2 - 4ac > 0$, then both roots are real
- If $b^2 - 4ac = 0$, then both roots are $-b/2a$
- If $b^2 - 4ac < 0$, then both roots are imaginary

Write a function $q(t)$ that takes a tuple $t = (a, b, c)$ and returns 1 if the roots are real, 0 otherwise:

$$q((a,b,c)) = \begin{cases} 1 & \text{roots are real} \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

For example,

$$q((1,4,-21)) = \text{True} \tag{25}$$

$$q((3,6,10)) = \text{False} \tag{26}$$

$$q((1,0,-4)) = \text{True} \tag{27}$$

Deliverables for Problem 2

- Review integers and reals.
- You are not allowed to use the module `cmath`.
- Complete the function.

Problem 3: Somethings Are Taxed, Somethings Are Not

Search on this phrase “what food isn’t taxed in indiana”. You’re writing software that allows for customer checkout at a grocery store. A customer will have a receipt r which is a list of pairs $r = [[i_0, p_0], [i_1, p_1], \ldots, [i_n, p_n]]$ where i is an item and p the cost. You have access to a list of items that are not taxed $no\_tax = [j_0, j_1, \ldots, j_m]$. The tax rate in Indiana is 7%. Write a function ’amt’ that takes the receipt and and the items that are not taxed and gives the total amount owed.

For this function you will have to implement the member function $m(x, lst)$ that returns True if x is a member of lst. For instance, this function can be used to check if an item (x) is present in the list (lst) of non taxable items, that will help to calculate the taxes accordingly.

For example, let $r = [[1, 1.45], [3, 10.00], [2, 1.45], [5, 2.00]]$ and $no\_tax = [33, 5, 2]$. Then

$$\begin{aligned} amt(r, no\_tax) &= round(((1.45 + 10.00)1.07 + 1.45 + 2.00), 2) && (28) \\ &= \$15.7 && (29) \end{aligned}$$

Deliverables for Problem 3

- Complete the function
- for $m(x, lst)$ you must search for x in lst by looping i.e., you are **not allowed** to use Python’s **in** keyword to check if an element exist inside a list. Instead, you should loop through the list and check it’s content.

Problem 4: Building a line in 2D

A line in 2D Euclidean space is given by $y = mx + b$ ('m' and 'b' are the slope and intercept respectively). Write a function f that takes two points $p_0 = (x_0, y_0), p_1 = (x_1, y_1)$, and returns the tuple (m, b):

$$f((x_0, y_0), (x_1, y_1)) = \begin{cases} (m, b) & x_0 \neq x_1 \\ () & \text{otherwise} \end{cases} \tag{30}$$

For example,

$$f((2,3),(6,4)) = (0.25, 2.5) \tag{31}$$

$$f((1,6),(3,2)) = (-2.0, 8.0) \tag{32}$$

$$f((1,3),(1,5)) = () \tag{33}$$

Deliverables for Problem 4

- Complete the function.

Problem 5: Means

When analyzing data, we often want to summarize it: make it concise. Each of these functions takes a list nlst of numbers. The mean of a list of numbers gives a summary through one number. You're probably aware of the arithmetic mean:

$$\text{arithmetic\_mean(nlst)} = \frac{(x_0 + x_1 + \cdots + x_{n-1})}{n} \tag{34}$$

For example, the arithmetic mean of [1,2,3] is 2.0.

The geometric mean (usually done with logs) is:

$$\text{geo\_mean(nlst)} = a^{sum/n} \tag{35}$$

$$sum = \log_a(x_0) + \log_a(x_1) + \cdots + \log_a(x_{n-1}) \tag{36}$$

where $\log_a$ is an arbitrary log to base a. For example, the geometric mean of [2,4,8] is 4.0. Use $\log_{10}$ as default–but it doesn't actually matter. The harmonic mean is:

$$\text{har\_mean(nlst)} = \frac{n}{1/x_0 + 1/x_1 + \cdots + 1/x_{n-1}} \tag{37}$$

For example, the harmonic mean of [1,2,3] is approximately 1.64.

The root mean square is:

$$\text{RMS\_mean(nlst)} = \sqrt{\frac{sum}{n}} \tag{38}$$

$$sum = x_0^2 + x_1^2 + \cdots + x_{n-1}^2 \tag{39}$$

For example, the root mean square of [1,3,4,5,7] is approximately 4.47.

All of these functions take a (possibly empty) list of numbers. If there is a list of numbers, then return the mean.

- If the list is empty, return the string, **Data Error: 0 values**
- If there is a zero in the list of numbers for the geometric or harmonic mean, then return the string **Data Error: 0 in data**

To help codify the problem, take a look at the unit testing and starter code. In both the test case file and starter code, we give these two errors and show them appropriately, because we face division by zero if we don't.

Deliverables for Problem 5

- Complete the functions as specified above.
- You can not use the python's **in** keyword to search for 0s in the list.
- Round the return values to two decimal places.
- **Do not change/edit** the error messages. Use them as such.

Problem 6: Cost Function

Suppose AirPure, a manufacturer of air filers, has a monthly fixed cost of $10,000

$$F(x) = \$10,000 \tag{40}$$

and a variable cost of $-0.0001x^2 + 10x$ for $0 \leq x \leq 40,000$ where x denotes the number of filters manufacturer per month

$$V(x) = \$ - 0.0001x^2 + 10x \tag{41}$$

Total cost C is the sum of variable and fixed cost:

$$C(x) = V(x) + F(x) \tag{42}$$

For example,

$$C(0) = 10000.0 \tag{43}$$
$$C(100) = 10999.0 \tag{44}$$
$$C(1000) = 19900.0 \tag{45}$$

Deliverables for Problem 6

- Complete the three functions.
- Hint: Read the Introduction again to get some help on this problem.

Problem 7: Mortgage

A *mortgage* is what you pay when you cannot purchase, usually a home, outright. You pay in installments that have to do with *terms* of the agreement. This includes an interest rate that is added to your payments. Here is the formula for your monthly payment:

$$m = P\frac{i(1+i)^n}{(1+i)^n - 1}$$

where P is the cost of the home, i is the percentage over 12 months, n is the total number of months. Let's say our data is: $300,000 house at 2.9% for 30 years. Then

$$\begin{aligned}
m &= 300000\frac{.029/12(1+.029/12)^{30\times 12}}{(1+.029/12)^{30\times 12}-1} \\
&= 300000\frac{.0024166(1.0024166)^{360}}{(1.0024166)^{360}-1} \\
&= 300000\frac{.0024166(2.38440696)}{2.38440696-1} \\
&= 300000\frac{0.005762}{1.38440696} \\
&= 300000(0.004162185) = \$1248.69/mo
\end{aligned}$$

The data should be in a list, *i.e.*, house = [300000,2.9,30] which represents the value of the house, the interest rate and the years. If house = [100000,6.0,30], the monthly payment is about $599.95. We will call this function Mortgage (house)

This seems easy financially. You can find what the mortgage *actually* costs by finding what you paid for the house and what its original value was:

$$\$1248.69/mo(30\,yr)(12\;mo/yr) - \$300000$$

$$\$449528.40 - \$300000 \approx \$149528.40$$

We will call this function total_paid(house).

As of today, the increasing of the federal interest rate has made purchasing home through a mortgage over 40% more expensive. This will be reflected in vehicles and credit cards too.

Deliverables for Problem 7

- Complete the functions.
- The total_paid function must use the Mortgage function. Both take lists described above
- Round the return values to two decimal places.

Problem 8: Geometric Series

A geometric series is a sequence of non-zero numbers in which each subsequent term is computed by multiplying the previous term by a constant non-zero number called the common ratio. The sequence can be described thusly:

$$a, ar, ar^2, ar^3, \ldots \tag{46}$$

where r is the common ratio. For any two successive terms we have:

$$\frac{s_{i+1}}{s_i} = \frac{ar^{i+1}}{ar^i} = r \tag{47}$$

We only need the first two values of a geometric sequence to generate it. In this problem, you'll implement a function that takes the first two values of a geometric sequence and a non-negative number and produces a list of the corresponding geometric sequence including the first two. For example,

$$geo([1, -3], 4) = [1, -3, 9, -27] \tag{48}$$

$$geo([10, 5], 4) = [10, 5, 2.5, 1.25] \tag{49}$$

$$geo([\sqrt{2}, -\sqrt{2}], 4) = [\sqrt{2}, -\sqrt{2}, \sqrt{2}, -\sqrt{2}] \tag{50}$$

Deliverables for Problem 8

- Complete the function
- Assume none of the numbers are zero

Problem 9: Looping

This is multiple part problem. Each problem helps you develop your skills in building solutions with loops. We will explicitly tell you whether to use an itegrator or subscription.

A: Smallest Two

Given a list of numbers lst, the function min_two returns the two smallest numbers. The smallest of the two numbers is the first item of the list. For example,

$$min\_two([5,4,3,2,1]) = [1,2] \tag{51}$$
$$min\_two([1,4,2,0,1,100] = [0,1] \tag{52}$$
$$min\_two([5,0,0,5] = [0,0] \tag{53}$$

B: Maximum Value(s)

Given a possibly empty list of numbers, determine the maximum value. If the list is empty, return the empty list. Since the maximum might not be unique return [x y] if the list has at least one value where x is the maximum and y the number of times it's in the list. You cannot use in built-in list functions. The function is called mm. For example,

$$mm([]) = [] \tag{54}$$
$$mm([1]) = [1,1] \tag{55}$$
$$mm([2,1,2,1,2]) = [2,3] \tag{56}$$
$$\tag{57}$$

C: Monotonicity

Given a list of numbers with at least one value, return true if the sequence is monotonic and false otherwise. A sequence of numbers $s_0, s_1, s_2, \ldots, s_n$ is monotonic if for any numbers s_i, the following number s_{i+1} is greater than or equal to the previous number *i.e.*, $s_i \leq s_{i+1}$. For example,

$$mo([1]) = true \tag{58}$$
$$mo([1, 1.1, 1.1, 1.3, 2]) = true \tag{59}$$
$$mo([20, 21, 22, 23, 22, 24]) = false \tag{60}$$
$$\tag{61}$$

D: Collegiate Wrestling Weight Classes

There are ten collegiate weight classes in the U.S. are:

$$[125, 133, 141, 149, 157, 165, 174, 184, 197, \text{"HW"}] \tag{62}$$

(there's a slight condition on heavy weight that we'll ignore for now). Assume a wrestler wants to know the heaviest weights that he is eligible for. Given a weight, return the list of weights that, in theory, can be wrestled. Heavy weight, the last value implied will be treated as a string. For example,

$$classes = [125, 133, 141, 149, 157, 165, 174, 184, 197, \text{"HW"}] \tag{63}$$

$$ww(classes, 110) = [125, 133, 141, 149, 157, 165, 174, 184, 197, \text{"HW"}] \tag{64}$$

$$ww(classes, 163) = [165, 174, 184, 197, \text{"HW"}] \tag{65}$$

$$ww(classes, 198) = [\text{"HW"}] \tag{66}$$

E: Distance between Points

Assume we have two tuples $p_0 = (x_0, x_1, \ldots, x_n)$ and $p_1 = (y_0, y_1, \ldots, y_n)$. We can find the distance between using

$$dis(p_0, p_1) = \sqrt{(x_0 - y_0)^2 + (x_1 - y_1)^2 + \cdots + (x_n - y_n)^2} \tag{67}$$

$$= (\sum_{i=0}^{n}(x_i - y_i)^2)^{1/2} \tag{68}$$

For example,

$$dis((1,2,3,1),(4,2,3,2)) = \sqrt{-3^2 + 0^2 + 0^2 + -1^2} \approx 3.16 \tag{69}$$

$$dis((1,2),(3,1)) = \sqrt{-2^2 + 1^2} \approx 2.24 \tag{70}$$

$$dis((3,),(2,)) = \sqrt{1^2} = 1 \tag{71}$$

Note: For distance, round your answer to 2 decimal places.

F: Trip

Assume you have a list of points $[p_0, p_1, p_2, \ldots, p_n]$ where $n \geq 1$. Using the function dis from above find the sum from p_0 to p_1 to $\ldots p_n$. If the list has only one point, then the distance is zero. For example,

$$trip([(1,),(3,),(7,)]) = dis((1,),(3)) + dis((3),(7)) = 6.0 \tag{72}$$

$$trip([(1,1)]) = 0 \tag{73}$$

$$trip([(0,0),(1,0),(1,1),(1,2)]) = dis((0,0),(1,0)) + \tag{74}$$

$$dis((1,0),(1,1)) + dis((1,1),(1,2)) = 3.0 \tag{75}$$

$$trip([(0,0,0),(1,1,1)]) = 1.73 \tag{76}$$

Deliverables for Problem 9

- Complete the functions.
- **A: Smallest Two**
  - Implement this as an iterator.
  - The list will always have at least two numbers.
- **B: Maximum Value(s)**
  - Implement this using subscripting.
  - You cannot use any built-in function like max

  .
- **C: Monotonicity**
  - Implement this as an iterator

  .
- **D: Wrestling Classes**
  - It's your choice whether to use iterator or subscripting. One approach will make the problem easier to code.
  - **Important** The list is always sorted in ascending order.
- **E: Distance**
  - It's your choice whether to use iterator or subscripting. One approach will make the problem easier to code.
  - Round to two places.
- **F: Trip**
  - It's your choice whether to use iterator or subscripting. One approach will make the problem easier to code.
  - Round to two places.
  - You must use the function in problem E i.e., 'dis'.

Student pairs

rghafoor@iu.edu, aketcha@iu.edu, eweidne@iu.edu
adamshm@iu.edu, blakruss@iu.edu
dadeyeye@iu.edu, vkethine@iu.edu
aaher@iu.edu, jolindse@iu.edu
omakinfi@iu.edu, skrasher@iu.edu
shakolia@iu.edu, jschlaef@iu.edu
abalbert@iu.edu, tdonoho@iu.edu
megalbin@iu.edu, jackssar@iu.edu
waasali@iu.edu, branwade@iu.edu
ahmalman@iu.edu, tfriese@iu.edu
anders14@iu.edu, jpenrigh@iu.edu
nsantoin@iu.edu, chataway@iu.edu
begaris@iu.edu, samstuar@iu.edu
jaybaity@iu.edu, dazamora@iu.edu
ianbaker@iu.edu, jchobbs@iu.edu
nbalacha@iu.edu, aubhighb@iu.edu
aiballou@iu.edu, btao@iu.edu
jabarbu@iu.edu, adhichin@iu.edu
cmbeaven@iu.edu, rjjorge@iu.edu
olibelch@iu.edu, nysach@iu.edu
jadbenav@iu.edu, leghuang@iu.edu
sberck@iu.edu, eliantu@iu.edu
evberg@iu.edu, jawashi@iu.edu
sbi@iu.edu, dmetodie@iu.edu
obianco@iu.edu, isclubia@iu.edu
jbilbre@iu.edu, sampopek@iu.edu
aibitner@iu.edu, zwoolley@iu.edu
jetblack@iu.edu, ertrice@iu.edu
ablashe@iu.edu, emcgough@iu.edu
pblasio@iu.edu, azaporo@iu.edu
dboecler@iu.edu, halejd@iu.edu
obowcott@iu.edu, nmccarry@iu.edu
sabowe@iu.edu, zaschaff@iu.edu
abrandtb@iu.edu, carcast@iu.edu
owebrook@iu.edu, stebutz@iu.edu
browpr@iu.edu, harpebr@iu.edu
ttbrowne@iu.edu, cgoeglei@iu.edu
ecaggian@iu.edu, cmcclar@iu.edu
petcarmi@iu.edu, lyncsara@iu.edu
jlcarrie@iu.edu, namcbrid@iu.edu

chenjunx@iu.edu, rorshiel@iu.edu
tchigudu@iu.edu, emiclar@iu.edu
scclotea@iu.edu, albperez@iu.edu
bbcolon@iu.edu, gmeinerd@iu.edu
jconcial@iu.edu, derthach@iu.edu
lizcoro@iu.edu, chaleas@iu.edu
lcosens@iu.edu, nfarhat@iu.edu
giancost@iu.edu, wodmaxim@iu.edu
bcrick@iu.edu, snsung@iu.edu
cwcrotty@iu.edu, jthach@iu.edu
mattcrum@iu.edu, kpmorse@iu.edu
pcullum@iu.edu, notsolo@iu.edu
tdearbor@iu.edu, jonhick@iu.edu
edeporte@iu.edu, uzrivera@iu.edu
jacdick@iu.edu, wgranju@iu.edu
dixonjh@iu.edu, zhangjoe@iu.edu
adolata@iu.edu, jonllam@iu.edu
maldowde@iu.edu, askrilof@iu.edu
ecdruley@iu.edu, wilsdane@iu.edu
majdunc@iu.edu, spletz@iu.edu
wjduncan@iu.edu, lufayshi@iu.edu
aareads@iu.edu, ellhuds@iu.edu
ebya@iu.edu, goel@iu.edu
seckardt@iu.edu, davthorn@iu.edu
gavedwar@iu.edu, vsivabad@iu.edu
ceifling@iu.edu, cjvanpop@iu.edu
augeike@iu.edu, ryou@iu.edu
jjepps@iu.edu, sharpky@iu.edu
jfahrnow@iu.edu, howelcar@iu.edu
chafiel@iu.edu, agesas@iu.edu
riflemin@iu.edu, patevig@iu.edu
foxjust@iu.edu, tanaud@iu.edu
jofuen@iu.edu, kninnema@iu.edu
magacek@iu.edu, rvu@iu.edu
landgarr@iu.edu, jomayode@iu.edu
dgodby@iu.edu, shawwan@iu.edu
gonzavim@iu.edu, ntatro@iu.edu
eg8@iu.edu, amurli@iu.edu
mahgree@iu.edu, avincelj@iu.edu
krgrohe@iu.edu, chrinayl@iu.edu
jgruys@iu.edu, gbharlan@iu.edu

kegupta@iu.edu, parksdr@iu.edu
kusgupta@iu.edu, bzurbuch@iu.edu
dgusich@iu.edu, jnjeri@iu.edu
rhaghver@iu.edu, nakoon@iu.edu
jhaile@iu.edu, istorine@iu.edu
hallzj@iu.edu, benrmitc@iu.edu
thamed@iu.edu, gmhowell@iu.edu
alehami@iu.edu, sehpark@iu.edu
hardenja@iu.edu, eawidema@iu.edu
rilmhart@iu.edu, abdjimoh@iu.edu
brohelms@iu.edu, aysiddiq@iu.edu
hernaga@iu.edu, rkabra@iu.edu
lohernan@iu.edu, kyrhod@iu.edu
mijherr@iu.edu, petersgm@iu.edu
bhmung@iu.edu, mooralec@iu.edu
phoen@iu.edu, limingy@iu.edu
nichhoff@iu.edu, aokhiria@iu.edu
tahoss@iu.edu, sarmayo@iu.edu
mihough@iu.edu, conthom@iu.edu
thuhtoo@iu.edu, pyahne@iu.edu
milahusk@iu.edu, zekerobe@iu.edu
bizzo@iu.edu, jscrogha@iu.edu
mahajenk@iu.edu, noramsey@iu.edu
bj13@iu.edu, joluca@iu.edu
oakagzi@iu.edu, owinston@iu.edu
skalivas@iu.edu, keysa@iu.edu
kekang@iu.edu, daknecht@iu.edu
brkapla@iu.edu, sousingh@iu.edu
rdkempf@iu.edu, samsieg@iu.edu
nekern@iu.edu, martiro@iu.edu
rkkhouri@iu.edu, oschwar@iu.edu
drewkimb@iu.edu, perkcaan@iu.edu
mkirolos@iu.edu, jurzheng@iu.edu
arkirt@iu.edu, stefschr@iu.edu
nolknies@iu.edu, cadwinin@iu.edu
abvekoes@iu.edu, agvore@iu.edu
arykota@iu.edu, ttsegai@iu.edu
fdkussow@iu.edu, nrs5@iu.edu
nkyryk@iu.edu, xujack@iu.edu
aalesh@iu.edu, arnpate@iu.edu
lewiserj@iu.edu, maxmuens@iu.edu

dl61@iu.edu, comojica@iu.edu
vimadhav@iu.edu, svuppunu@iu.edu
mccoyry@iu.edu, jamundy@iu.edu
kmcinto@iu.edu, kereidy@iu.edu
skmcmaho@iu.edu, abramjee@iu.edu
luilmill@iu.edu, msisodiy@iu.edu
mitcchar@iu.edu, johsong@iu.edu
morrmaja@iu.edu, kyeosen@iu.edu
jnmroch@iu.edu, jensprin@iu.edu
alenmurp@iu.edu, elyryba@iu.edu
wemurray@iu.edu, msmelley@iu.edu
joelna@iu.edu, bwinckle@iu.edu
shnaka@iu.edu, mr86@iu.edu
sndashi@iu.edu, bzurbuch@iu.edu
davingo@iu.edu, tcpatel@iu.edu
laynicho@iu.edu, cmw26@iu.edu
anuttle@iu.edu, stusinha@iu.edu
gokeefe@iu.edu, antreye@iu.edu
jtohland@iu.edu, ayuraiti@iu.edu
coolds@iu.edu, ereno@iu.edu
lanounch@iu.edu, as145@iu.edu
jpascov@iu.edu, kt10@iu.edu
patelkus@iu.edu, csmalarz@iu.edu
patel88@iu.edu, envu@iu.edu
mmpettig@iu.edu, ashmvaug@iu.edu
rpogany@iu.edu, evmtaylo@iu.edu
phiprice@iu.edu, nasodols@iu.edu
etprince@iu.edu, esisay@iu.edu
jarabino@iu.edu, mwroark@iu.edu
csradtke@iu.edu, matzhang@iu.edu
rraguram@iu.edu, lvanjelg@iu.edu
marebey@iu.edu, egshim@iu.edu
kreddiva@iu.edu, ianwhit@iu.edu
sunreza@iu.edu, darisch@iu.edu
rogerju@iu.edu, owasmith@iu.edu
jbromers@iu.edu, jeffsung@iu.edu
jrosebr@iu.edu, ptstorm@iu.edu
ssetti@iu.edu, rtrujill@iu.edu
burshell@iu.edu, rsstarli@iu.edu
mysoladi@iu.edu, awestin@iu.edu
samsteim@iu.edu, joeywill@iu.edu

jttrinkl@iu.edu, bdzhou@iu.edu
sjvaleo@iu.edu, tavalla@iu.edu
mvanworm@iu.edu, zhaofan@iu.edu
kviele@iu.edu, jwa14@iu.edu
nmwaltz@iu.edu, mew17@iu.edu

Honors section pairs

ethcarmo@iu.edu, venguyen@iu.edu
ligonza@iu.edu, aktumm@iu.edu, swa5@iu.edu
nagopi@iu.edu, marafoth@iu.edu
zfhassan@iu.edu, jarenner@iu.edu
tkefalov@iu.edu, nzaerhei@iu.edu
alindval@iu.edu, snresch@iu.edu
joshprat@iu.edu, arangwan@iu.edu
avreddy@iu.edu, lorivera@iu.edu
huntang@iu.edu, sturaga@iu.edu