C200 Programming Assignment № 10: Classes, SQL, Random Walks

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

Indiana University, Bloomington, IN, USA

December 2, 2022

Introduction

- Programming partners are present at the end of this document.
- **Due Date** Friday, December 9, 11:00PM EST.
- You should submit to Autograder (`c200.luddy.indiana.edu`) and merely pushing to GitHub is not enough.

Instructions for submitting to the Autograder (`c200.luddy.indiana.edu`)

1. Make sure that you are **following the instructions** in the PDF, especially the format of output returned by the functions. For example, if a function is expected to return a numerical value, then make sure that a numerical value is returned (not a list or a dictionary). Similarly, if a list is expected to be returned then return a list (not a tuple, set or dictionary).
2. Test **debug** the code well (syntax, logical and implementation errors), before submitting to the Autograder. These errors can be easily caught by running the code in VSC and watching for unexpected behavior such as, program failing with syntax error or not returning correct output.
3. Check that the **code does not have infinite loop (that never exits) or an endless recursion (that never completes)** before submitting to the Autograder. You can easily check for this by running in VSC and watching for program output, if it terminates timely or not.
4. **Remove** ir-relavant library imports that are not explicitly allowed by the HW. For exmaple, if we did not use the library 'tkinter' then please don't import it in the code.
5. Given that you already tried above points (points 1-4), if you see that Autograder does not do anything (after you press 'submit') and waited for a while (30 seconds to 50 seconds), try refreshing the page or using a different browser.
6. Once you are done testing your code, comment out the tests i.e. the code under the __name__ == "__main__" section.

Problem 1: Random Walk

A random walk is a *stochastic process*. A stochastic process is a series of values that are not determined functionally, but probabilistically. The random walk is supposed to describe an inebriated person who, starting from the bar, intends to walk home, but because of intoxication instead randomly takes single steps either forward or backward, left or right. The person has no memory of any steps taken, so theoretically, the person shouldn't move too far from where he or she starts. Random walks are used to model many phenomena, like the size of the web or changes in financial instruments. We will model a 2D random walk with two arrays x and y where x represents moving left or right and y represents forward or backward. The index i will represent the step and x[i],y[i] will represent the location at step i. So, for i=0, we have x[0],y[0] (starting place). Using random we choose from the list [1,2,3,4]. If the value is one then we move right from the previous x position:

```
x[i] = x[i-1] + 1
y[i] = y[i-1]
```

If the value is two, then we move left from the previous x position:

```
x[i] = x[i-1] - 1
y[i] = y[i-1]
```

If the value is three, we move up from the previous y position:

```
x[i] = x[i-1]
y[i] = y[i-1] + 1
```

And when the value is four, we move down from the previous y position:

```
x[i] = x[i-1]
y[i] = y[i-1] - 1
```

Here is another way to describe this:

$$step(0) = 0 \tag{1}$$

$$step_i(n) = \begin{cases} \text{right} & step(n-1),\ i=1 \\ \text{left} & step(n-1),\ i=2 \\ \text{up} & step(n-1),\ i=3 \\ \text{down} & step(n-1),\ i=4 \end{cases} \tag{2}$$

Although the example above may seem like recursion but **this is not recursion**. You are not required to use recursion to solve this problem.

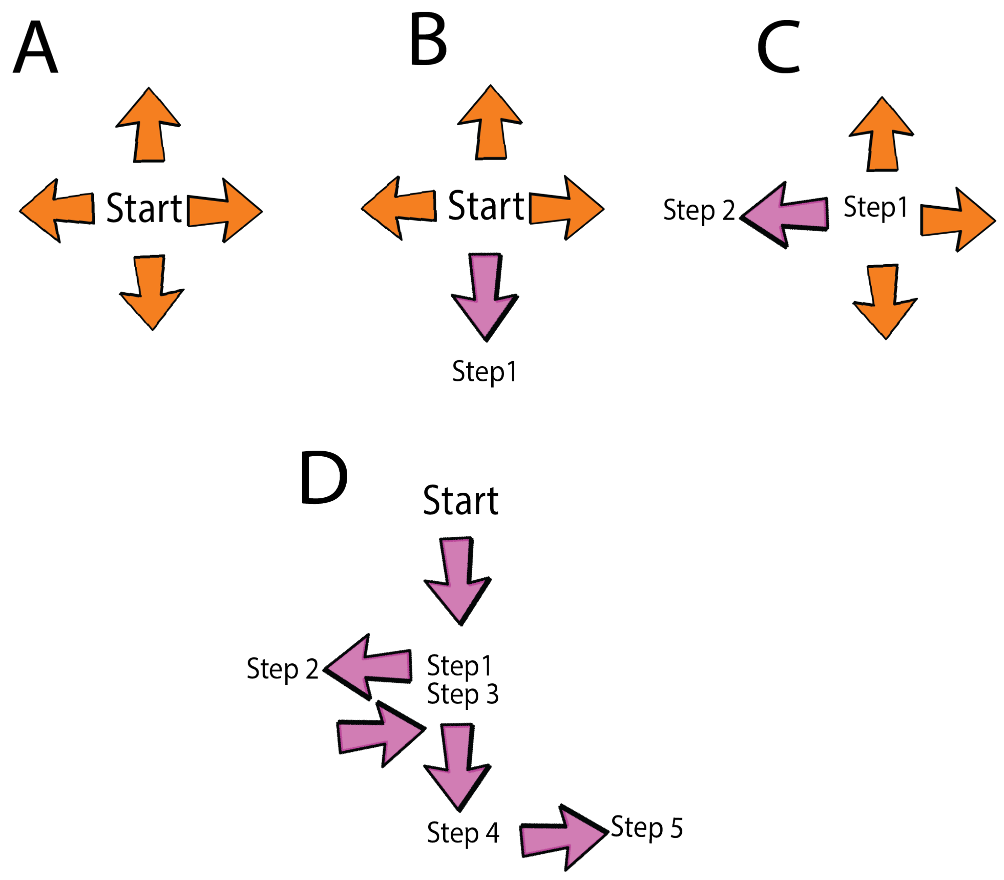

Figure 1: The first few steps of a random walk.

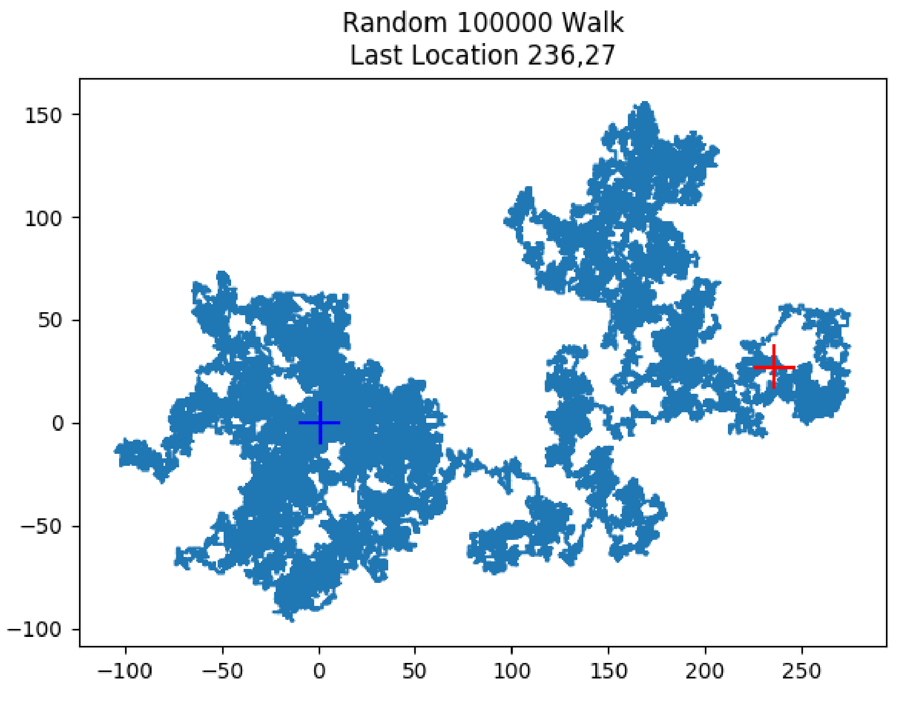

Figure 2: The blue cross is where we started and the red cross is where we ended. Since each run is random, and we have 100,000 steps, your plot will not look similar to mine.

Programming Problem 1: Random Walk

- Complete the `step` function.
- These are *random walks*; hence, the outputs will be different .

Problem 2: Vector Class

In this problem, you'll implement a mathematical vector class. While the vectors are generally depicted as columns, you'll use a tuple. A vector of size $n \in \{1, 2, 3, \ldots\}$ is a collection of real numbers:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{3}$$

Given $\mathbf{x}, \mathbf{y}$ are vectors of size n and z is a real number-we can define the following operations

$$\mathbf{x} + \mathbf{y} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_n + y_n \end{bmatrix} \tag{4}$$

$$\mathbf{x} - \mathbf{y} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_1 - y_1 \\ x_2 - y_2 \\ \vdots \\ x_n - y_n \end{bmatrix} \tag{5}$$

$$\mathbf{x} \times \mathbf{y} = \sum_{i=1}^{n} x_i y_i \tag{6}$$

$$z \times \mathbf{x} = z \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} z \times x_1 \\ z \times x_2 \\ \vdots \\ z \times x_n \end{bmatrix} \tag{7}$$

$$-\mathbf{x} = \begin{bmatrix} -x_1 \\ -x_2 \\ \vdots \\ -x_n \end{bmatrix} \tag{8}$$

$$|\mathbf{x}| = abs(\mathbf{x}) = \sqrt{\mathbf{x} \times \mathbf{x}} \tag{9}$$

(10)

Two vectors $\mathbf{x}, \mathbf{y}$ are equal if $x_i = y_i$ for $1 \leq i \leq n$. The class you'll build will work for any non-zero length vector. The class constructor is:

```
class Vector:
    def __init__(self, *x):
        self.__v = x
```

The *x means you are making the arguments a tuple–further, a tuple can be sent if it has *. For example,

```
w = Vector(*(10,10))
```

will send the tuple as a tuple–this makes arguments easier. For a vector $\mathbf{x}$ of length n, print($\mathbf{x}$) should display

```
<x1,x2,...,xn>
```

Here are instances and outputs:

```
x,y,w = Vector(1,2),Vector(3,-1),Vector(*(10,10))
z,a = Vector(0,3,2),Vector(-1,-1,-1)
print(x,y,z,a)
print(x+y,z+a)
print(x*y,z*a)
print(5*x,5*z)
print(abs(x),abs(z))
print(-x,-z)
print(x - y + y == x, 2 * z == z + z)
```

with output:

```
<1, 2> <3, -1> <0, 3, 2> <-1, -1, -1>
<4, 1> <-1, 2, 1>
1 -5
<5, 10> <0, 15, 10>
2.23606797749979 3.605551275463989
<-1, -2> <0, -3, -2>
True True
```

Deliverable for Problem 2: Vectors

- Revising the concept of object-oriented programming from lectures/labs may come in handy while working on this problem.
- Complete the class–use a tuple to represent the mathematical vector.
- For vector multiplication $(\mathbf{x} \times \mathbf{y})$ and abs() function, round the answer to 3 decimal places.
- You can use Python's sum function.
- You are free to use any Python string class method.
- I have completed a couple of the methods. For the remaining, unfinished ones decide whether you need to return the vector type or another type.

Queries

In class we were introduced to SQL and the relational model. You will have a great deal of freedom with this problem. Create a table called `Weather` with attributes `City`, `State`, `High`, `Low` and populate it with the data shown in Table 1. I've made equivalent list comprehension on

Weather

City	State	High	Low
Phoenix	Arizona	105	90
Tucson	Arizona	101	92
Flag Staff	Arizona	105	90
San Diego	California	77	60
Albuquerque	New Mexico	80	72
Nome	Alaska	64	-54

Table 1: Relation Weather and tuples

.

this iterable:

```python
data = [('Phoenix', 'Arizona', 105, 90),
           ('Tucson', 'Arizona', 101, 92),
           ('Flag Staff', 'Arizona', 105, 90),
           ('San Diego', 'California', 77, 60),
           ('Albuquerque', 'New Mexico', 80, 72),
           ('Nome', 'Alaska', 64 ,-54)]
```

You should read about these SQL functions (**NOTE**: SQL functions, not Python functions):

count(), sum(), min(), max() as well as "group by" and "in". We've given the answer to query 1 (i.e. MYSQL query). The results of these queries are shown in the output. The following section shows the results obtained by using python and list comprehension. Since LC and SQL are very similar, the LC might give you some ideas on what to do. Since there's no looping in SQL, you'll have to simplify your solutions. You are required to implement the SQL operations in python to implement these queries.

1. Select all the tuples (Query 1)

```
print("Query 1")
    for i in my_cursor.execute("SELECT * FROM Weather"):
        print(i)
print("List Comprehension: ", data)
```

2. Select all the tuples where the High temperature is less than 80 (Query 2)

```
print("Query 2")
print("List Comprehension: ", [d for d in data if d[2] < 80 ])
```

3. Select All the cities where the low temperature is strictly greater than the Low of Albuquerque – you cannot use the number 72.0 in the query (Query 3)

```
print("Query 3")
x =[d[0] for d in data if d[3] > [d[3] for d in data if d[0] == '↩
    Albuquerque'][0]]
print("List Comprehension: ",x)
```

4. Select the city and temperature with the smallest low temperature (Query 4)

```
print("Query 4")
print("List Comprehension: ",[(d[0],d[3]) for d in data if d[3] in (↩
    sorted(data, key = lambda x:x[3])[0])])
```

5. Select the city temperature with the largest high temperature–since there are two, both cities should be returned. (Query 5)

```
print("Query 5")
print("List Comprehension: ",[(d[0],d[2]) for d in data if d[2] in (↩
    sorted(data, key = lambda x:x[2],reverse=True)[0])])
```

6. Display the *average* High and Low temperatures–you are not allowed to use Avg() (Query 6)

```python
print("Query 6")
print("List Comprehension: ", [(sum([d[2] for d in data])/len(data),↩
    sum([d[3] for d in data])/len(data))])
```

7. Give the counts of cities by their Low temperatures (Query 7)

```python
print("Query 7")
print([(i,list(map((lambda x: x[3]),data)).count(i)) for i in set(map↩
    ((lambda x: x[3]),data))])
```

Output

```
Query 1
('Phoenix', 'Arizona', 105.0, 90.0)
('Tucson', 'Arizona', 101.0, 92.0)
('Flag Staff', 'Arizona', 105.0, 90.0)
('San Diego', 'California', 77.0, 60.0)
('Albuquerque', 'New Mexico', 80.0, 72.0)
('Nome', 'Alaska', 64.0, -54.0)
List Comprehension:
[('Phoenix', 'Arizona', 105, 90),
 ('Tucson', 'Arizona', 101, 92),
 ('Flag Staff', 'Arizona', 105, 90),
 ('San Diego', 'California', 77, 60),
 ('Albuquerque', 'New Mexico', 80, 72),
 ('Nome', 'Alaska', 64, -54)]
Query 2
('San Diego', 'California', 77.0, 60.0)
('Nome', 'Alaska', 64.0, -54.0)
List Comprehension:
[('San Diego', 'California', 77, 60),
 ('Nome', 'Alaska', 64, -54)]
Query 3
('Phoenix',)
('Tucson',)
('Flag Staff',)
List Comprehension:  ['Phoenix', 'Tucson', 'Flag Staff']
```

Output

```
Query 4
('Nome', -54.0)
List Comprehension:  [('Nome', -54)]
Query 5
('Phoenix', 105.0)
('Flag Staff', 105.0)
List Comprehension:  [('Phoenix', 105), ('Flag Staff', 105)]
Query 6
(88.66666666666667, 58.333333333333336)
List Comprehension:  [(88.66666666666667, 58.333333333333336)]
Query 7
(-54.0, 1)
(60.0, 1)
(72.0, 1)
(90.0, 2)
(92.0, 1)
List Comprehension:  [(72, 1), (-54, 1), (60, 1), (90, 2), (92, 1)]
```

SQL

- Write the MYSQL code using python for queries 1-7.
- Note: To reiterate, you must first successfully create the table, then comment out the code for table creation or it'll throw an error. A query should be run after the table creation and populating it with data.
- To verify your output, you can cross-check it with the output produced by LC for each query.

Pairs

C200 Student Pairs adamshm@iu.edu, jbilbre@iu.edu
dadeyeye@iu.edu, derthach@iu.edu
aaher@iu.edu, vkethine@iu.edu
omakinfi@iu.edu, zhangjoe@iu.edu
shakolia@iu.edu, lizcoro@iu.edu
abalbert@iu.edu, jtohland@iu.edu
megalbin@iu.edu, jonhick@iu.edu
waasali@iu.edu, mahajenk@iu.edu
ahmalman@iu.edu, jensprin@iu.edu
anders14@iu.edu, hernaga@iu.edu
nsantoin@iu.edu, wgranju@iu.edu
jaybaity@iu.edu, dl61@iu.edu

ianbaker@iu.edu, comojica@iu.edu
nbalacha@iu.edu, brkapla@iu.edu
aiballou@iu.edu, aibitner@iu.edu
jabarbu@iu.edu, ttsegai@iu.edu
cmbeaven@iu.edu, landgarr@iu.edu
olibelch@iu.edu, cmw26@iu.edu
sberck@iu.edu, luilmill@iu.edu
evberg@iu.edu, skalivas@iu.edu
sbi@iu.edu, augeike@iu.edu
jetblack@iu.edu, rdkempf@iu.edu
ablashe@iu.edu, nmccarry@iu.edu
pblasio@iu.edu, jnjeri@iu.edu
dboecler@iu.edu, jamundy@iu.edu
obowcott@iu.edu, albperez@iu.edu
sabowe@iu.edu, wilsdane@iu.edu
abrandtb@iu.edu, bhmung@iu.edu
owebrook@iu.edu, hardenja@iu.edu
browpr@iu.edu, ashmvaug@iu.edu
ttbrowne@iu.edu, jacdick@iu.edu
stebutz@iu.edu, hallzj@iu.edu
ecaggian@iu.edu, askrilof@iu.edu
petcarmi@iu.edu, skmcmaho@iu.edu
jlcarrie@iu.edu, ellhuds@iu.edu
carcast@iu.edu, leghuang@iu.edu
chenjunx@iu.edu, shnaka@iu.edu
tchigudu@iu.edu, pcullum@iu.edu
adhichin@iu.edu, dazamora@iu.edu
scclotea@iu.edu, nolknies@iu.edu
jconcial@iu.edu, samsieg@iu.edu
lcosens@iu.edu, nichhoff@iu.edu
giancost@iu.edu, egshim@iu.edu
bcrick@iu.edu, envu@iu.edu
cwcrotty@iu.edu, martiro@iu.edu
mattcrum@iu.edu, jnmroch@iu.edu
tdearbor@iu.edu, zekerobe@iu.edu
edeporte@iu.edu, maldowde@iu.edu
dixonjh@iu.edu, rsstarli@iu.edu
adolata@iu.edu, kviele@iu.edu
tdonoho@iu.edu, notsolo@iu.edu
wjduncan@iu.edu, nkyryk@iu.edu
aareads@iu.edu, harpebr@iu.edu

ebya@iu.edu, coolds@iu.edu
seckardt@iu.edu, rghafoor@iu.edu
ceifling@iu.edu, patevig@iu.edu
jpenrigh@iu.edu, shawwan@iu.edu
jjepps@iu.edu, bj13@iu.edu
jfahrnow@iu.edu, elyryba@iu.edu
nfarhat@iu.edu, ayuraiti@iu.edu
chafiel@iu.edu, nasodols@iu.edu
riflemin@iu.edu, rogerju@iu.edu
magacek@iu.edu, gmeinerd@iu.edu
dgodby@iu.edu, daknecht@iu.edu
cgoeglei@iu.edu, nakoon@iu.edu
goel@iu.edu, joelna@iu.edu
gonzavim@iu.edu, jomayode@iu.edu
eg8@iu.edu, laynicho@iu.edu
mahgree@iu.edu, joluca@iu.edu
jgruys@iu.edu, davthorn@iu.edu
kegupta@iu.edu, tcpatel@iu.edu
kusgupta@iu.edu, rkabra@iu.edu
dgusich@iu.edu, patelkus@iu.edu
jhaile@iu.edu, gmhowell@iu.edu
halejd@iu.edu, aketcha@iu.edu
chaleas@iu.edu, petersgm@iu.edu
thamed@iu.edu, cadwinin@iu.edu
alehami@iu.edu, nysach@iu.edu
rilmhart@iu.edu, aubhighb@iu.edu
chataway@iu.edu, mwroark@iu.edu
brohelms@iu.edu, bdzhou@iu.edu
lohernan@iu.edu, mijherr@iu.edu
jchobbs@iu.edu, kninnema@iu.edu
phoen@iu.edu, blakruss@iu.edu
tahoss@iu.edu, azaporo@iu.edu
mihough@iu.edu, sndashi@iu.edu
howelcar@iu.edu, uzrivera@iu.edu
thuhtoo@iu.edu, spletz@iu.edu
bizzo@iu.edu, arkirt@iu.edu
rjjorge@iu.edu, wodmaxim@iu.edu
oakagzi@iu.edu, darisch@iu.edu
nekern@iu.edu, rvu@iu.edu
keysa@iu.edu, eweidne@iu.edu
drewkimb@iu.edu, oschwar@iu.edu

mkirolos@iu.edu, owasmith@iu.edu
abvekoes@iu.edu, alenmurp@iu.edu
fdkussow@iu.edu, msisodiy@iu.edu
aalesh@iu.edu, sarmayo@iu.edu
lewiserj@iu.edu, nmwaltz@iu.edu
limingy@iu.edu, jarabino@iu.edu
jolindse@iu.edu, nrs5@iu.edu
jonllam@iu.edu, skrasher@iu.edu
isclubia@iu.edu, svuppunu@iu.edu
lyncsara@iu.edu, emcgough@iu.edu
vimadhav@iu.edu, johsong@iu.edu
namcbrid@iu.edu, agvore@iu.edu
mccoyry@iu.edu, ssetti@iu.edu
kmcinto@iu.edu, as145@iu.edu
dmetodie@iu.edu, kyrhod@iu.edu
benrmitc@iu.edu, tanaud@iu.edu
kpmorse@iu.edu, abramjee@iu.edu
wemurray@iu.edu, lufayshi@iu.edu
davingo@iu.edu, ryou@iu.edu
anuttle@iu.edu, lanounch@iu.edu
gokeefe@iu.edu, ntatro@iu.edu
aokhiria@iu.edu, jwa14@iu.edu
kyeosen@iu.edu, rraguram@iu.edu
sehpark@iu.edu, sampopek@iu.edu
parksdr@iu.edu, bzurbuch@iu.edu
jpascov@iu.edu, antreye@iu.edu
patel88@iu.edu, rtrujill@iu.edu
perkcaan@iu.edu, jbromers@iu.edu
mmpettig@iu.edu, msmelley@iu.edu
rpogany@iu.edu, eawidema@iu.edu
phiprice@iu.edu, stefschr@iu.edu
etprince@iu.edu, mvanworm@iu.edu
csradtke@iu.edu, sunreza@iu.edu
marebey@iu.edu, jeffsung@iu.edu
kreddiva@iu.edu, bwinckle@iu.edu
ereno@iu.edu, mew17@iu.edu
mr86@iu.edu, istorine@iu.edu
jrosebr@iu.edu, zaschaff@iu.edu
jscrogha@iu.edu, vsivabad@iu.edu
sharpky@iu.edu, jttrinkl@iu.edu
burshell@iu.edu, avincelj@iu.edu

rorshiel@iu.edu, bzurbuch@iu.edu
sousingh@iu.edu, kt10@iu.edu
esisay@iu.edu, ertrice@iu.edu
csmalarz@iu.edu, jurzheng@iu.edu
mysoladi@iu.edu, samstuar@iu.edu
samsteim@iu.edu, matzhang@iu.edu
ptstorm@iu.edu, ianwhit@iu.edu
snsung@iu.edu, conthom@iu.edu
btao@iu.edu, cjvanpop@iu.edu
evmtaylo@iu.edu, lvanjelg@iu.edu
jthach@iu.edu, zhaofan@iu.edu
joeywill@iu.edu, zwoolley@iu.edu
xujack@iu.edu, pyahne@iu.edu

H200 Student Pairs

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