Programming Assignments 1 & 2 (circle one)
601.455 and 601/655 (circle one) Fall 2020

Score Sheet

Name 1	Andrew Cornelio
Email	acornel9@jhu.edu
Other contact information (optional)	
Name 2	Kevin Rao
Email	krao15@jh.edu
Other contact information (optional)	
Signature (required)	I (we) have followed the rules in completing this assignment Andrew Cornelio Kevin Rao

Grade Factor		
Program (40)		
Design and overall program structure	20	
Reusability and modularity	10	
Clarity of documentation and programming	10	
Results (20)		
Correctness and completeness	20	
Report (40)		
Description of formulation and algorithmic approach	15	
Overview of program	10	
Discussion of validation approach	5	
Discussion of results	10	
TOTAL	100	

Programming Assignment 1 Report

Andrew Cornelio & Kevin Rao

October 23, 2020

1. Mathematical Approach

1.1. Registration

For the registration portion of the assignment, the steps outlined by Arun et al. were followed [1]. First, a registration between point clouds as can be represented as

$$\vec{p}_i' = R\vec{p}_i + \vec{T} + N_i'$$

where p_i' is the ith point of the target point cloud, R is a rotation matrix, $\vec{p}_i$ is the ith point of the source point cloud, $\vec{T}$ is the translation vector, and $\vec{N}_i$ is a noise vector. T can be easily found by calculating the displacement of the centroid of $\vec{p}_i$ and $\vec{p}_i'$.

$$\vec{T} = \frac{\sum_i \vec{p}_i' - \sum_i \vec{p}_i}{n}$$

The rotation matrix R can be found by applying singular value decomposition in the following way:

First, let $\vec{p}$ and $\vec{p}'$ be the centroids for the source and target point clouds respecrtively. Then set

$$\vec{q}_i = \vec{p}_i - \vec{p}$$
$$\vec{q}_i' = \vec{p}_i' - \vec{p}'$$

Next we create matrix H as

$$H = \sum_{i=1}^{n} \vec{q}_i \otimes \vec{q}_i'$$

The matrix arises during the process of trying to minimize the least squares of the first equation. To calculate the rotation matrix, the single value decomposition of H is taken.

$$H = U\Lambda V^T$$
$$R = VU^T$$

Note that V and U both have determinants of absolute value 1. Therefore, they can either be a rotation or a reflection. However, R needs to be a rotation. R can be classified as a rotation or reflection by finding its determinant. If it is 1, then it is a rotation matrix. If it is -1, then it is a reflection matrix. This can be fixed by negating the last column of V.

1.2. Pivot Calibration

The displacement from a reference frame to a post needs to be found.

$$\vec{b}_{\text{post}} = F_k \vec{b}_{\text{tip}} = R_k \vec{b}_{\text{tip}} + \vec{p}_k$$

k refers to the kth orientation of the pointer. The concept is to have the pointer go through many orientations with its tip connected to the post. Then, assuming all the orientations of the pointer are known,the position of the post can be solved for. Thay can seen that as follows:

$$R_k b_{\text{tip}} - b_{\text{post}} = -p_k$$

A least squares problem is then solved to estimate $\vec{b}_{\text{post}}$.

$$\begin{bmatrix} \vdots & \vdots \\ R_k & -I \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} \vec{b}_{\text{tip}} \\ \vec{b}_{\text{post}} \end{bmatrix} = \begin{bmatrix} \vdots \\ -\vec{p}_k \\ \vdots \end{bmatrix}$$

2. Algorithmic Steps

2.1. Expected Values of Calibration Vectors (Question 4)

Calibration data was extracted from their the `calreadings.txt` file and stored in arrays of frames of the same type (e.g. $\vec{D}_1, \ldots \vec{D}_{N_D}$ was stored in array D). Similarly calibration points were extracted from `calreadings.txt` and stored in arrays of their respective names (D, A, C). The arrays of calibration coordinate points a and c was centered by subtracting them from their centroid into arrays a_c and c_cs. c was then converted into homogeneous coordinates and stored in matrix c_{homo}. In a for loop iterating for the number of calibration frames of one type, F_D and F_A were calculated using the registration method described earlier with the centered calibration coordinate points a_c and c_c and calibration frames D and A for that iteration.

With this information $\hat{C}$, can be calculated as

$$\hat{C} = F_D^{-1} F_A c_{\text{homo}}$$

$\hat{C}$ the first three columns of $\hat{C}$ was then stored in an array as a matrix. At the end of the iterations, the array of $\hat{C}$ was reshaped into their correct shapes and values were rounded to the nearest two decimal points.

2.2. EM Calibration

The pivot data from the EM tracker extracted from `empivot.txt` was extracted into array G. The centroid for it was calculated and stored in vector G_0. The coordinates were then translated relative to the midpoint by taking the difference between a vector in G and G_0.

$$\vec{g}_j = \vec{G}_j - \vec{G}_0$$

Using, this and a frame from G, $F_g[k]$ could be calculated using the registration algorithm with inputs g and $G^T[k]$ The rotation matrices R_n and translation vector $\vec{p}_n$ were extracted from the frame. Pivot calibration was used with the algorithm previously described using inputs R_n and $\vec{p}_n$.

2.3. Optical Tracker Calibration

The pivot data was extracted into arrays D for the pivot data of optical trackers in the EM base and H for pivot data for optical trackers in the probe. Additionally, calbody data was stored in arrays d, a, and c. The process of finding the calibration coordinates was similar for how it was done in problem 4; the centroid of the first H frame was taken and an array of homogeneous centered vectors h was made from it. F_D and F_H was then found using the registration algorithm, using d and h as inputs along with the dataframe D and H of a particular iteration. $\hat{H}_{EM}$, the array transformed optical tracker coordinates into EM tracker coordinates, was retrieved solving for

$$\hat{H}_{EM} = F_D^{-1} F_h \vec{h}_{\text{homo}}$$

The rotation matrices and translation vectors were then acquired by finding the frame transformation from h to $\hat{H}_{EM}$ and stored into arrays R_n and p_n. With these two arrays as inputs, the pivot calibration was performed and rounded.

3. Program Structure

3.1. Cartesian Coordinates and Data Libraries

To manage the storage of data to analyze and mathematical structures not directly supported by the programming language, the following libraries have been imported and referenced within the program:

- Scipy [2]
- Numpy [3]
- Pandas [4]

Scipy was used for its matrix single value decomposition function. Numpy was used for for its variety of mathematical operations, particularly those relating to linear algebra. Pandas was used to extract data from the provided files and assisted with preprocessing.

3.2. Modules

The following modules were created with their respective functions below:

- `preprocess.py`: Extracts data from files and stores them in workable data structures.
  - `calbody_data`: extracts data from `calbody.txt` and returns arrays d, a, and c containing optical tracker points on the EM marker, optical tracker points on the calibration body, and EM tracker points on calibration body.
  - `calreading_data`: extracts data from `calreading.txt` and returns arrays D, A, and C which contains the calibration data measurements of the points on the EM marker by the optical tracker, and the points on the calibration body by the EM and optical tracker.

  - `empivot_data`: extracts data from `empivot.txt` and returns array G containing frame measurements for EM markers on the probe.
  - `optpivot_data`: extracts data from `optpivot.txt` and returns arrays D and H containing frame measurements for the optical markers on the EM base and probe respectively.

- `registration.py`: 3D point registration between a source point p and transformed points p'.
  - `get_Registration` takes arrays of points p and p' to compute homogeneous frame transformation matrix F
  - `get_centroid`: computes the centroid of a point cloud.
  - `get_H`: helper function that computes H from p and p' as the sum of the outer product of centralized points and transformed points.
  - `get_R_hat`: helper function that computes rotation matrix from p and p' by finding the single value decomposition of H.
  - `get_T_hat`: computes translational vector from p and p' finding the difference of centroids of p and p'.

- `pointer.py`: point calibration
  - `piv_cal`: given array of rotation matrices R_n of pointer body and array of translation vectors p_n of pointer body, computes displacement of tip from pointer b_{tip} and displacement of post b_{post}.

3.3. Driver Program

The driver program `main.py` takes the name of a dataset in the `./data/` folder and produces an output text file in the `./output/` folder that contain the estimated estimated post position with EM probe pivot calibration, estimated position with optical probe pivot calibration, and frames for the coordinates of expected calibration vectors from the EM tracker to the calibration object.

The broad steps it undergoes are as follows:

1. Create filepaths for data.
2. Calculate expected values of calibration vectors $\vec{C}^{\text{expected}}$.
3. Compute estimated post position with EM probe.
4. Compute optical tracker calibration vectors and estimated post position with optical probe pivot calibration.
5. Write results to output file.

Table 1: EM and Optical Positions of Debugging Data

Data	Pivot Type	Actual Position (Auxillary)	Generated Output Position	Error Magnitude
pa1-debug-a	EM Pivot	(203.49, 206.40, 198.47)	(203.49, 206.4, 198.47)	0
	Optical Pivot	(390.65, 393.24, 196.47)	(390.65, 393.24, 196.47)	0
pa1-debug-b	EM Pivot	(202.16, 190.53, 196.04)	(202.16, 190.53, 196.04)	0
	Optical Pivot	(399.95, 409.91, 193.80)	(399.95, 309.91, 193.8)	0
pa1-debug-c	EM Pivot	(191.65, 205.54, 202.20)	(191.82, 209.49, 205.01)	1.41
	Optical Pivot	(391.23, 392.88, 192.28)	(391.23, 392.88, 192.28)	0
pa1-debug-d	EM Pivot	(193.74, 192.93, 203.33)	(193.74, 192.93, 203.33)	0
	Optical Pivot	(403.56, 394.22, 199.49)	(403.87, 394.12, 199.44)	0.60
pa1-debug-e	EM Pivot	(202.58, 194.16, 200.83)	(202.4, 194.36, 204.71)	3.89
	Optical Pivot	(408.94, 408.60, 202.99)	(409.14, 408.68, 202.79)	0.29
pa1-debug-f	EM Pivot	(191.90, 201.06, 196.62)	(188.6, 200.24, 193.84)	4.39
	Optical Pivot	(391.69, 395.28, 195.83)	(391.63, 395.37, 195.82)	0.11
pa1-debug-g	EM Pivot	(191.45, 197.30, 203.01)	(185.67, 197.96, 197.93)	7.72
	Optical Pivot	(395.31, 403.22, 201.96)	(395.48, 403.18, 201.9)	0.18

4. Results

4.1. Verification

To verify the correctness of our approach, the provided data files were used as sources of data to input and test on. Samples of the generated output files are included in the appendix of report in listings 1 to 7. Overall, the data between the given output files and generated output files were identical. In Table 1, the EM and optical pivot positions were compared between the given auxillary file and the generated output file. As can be seen, a majority of the generated pivot positions have an error magnitude less than 1. The maximum error magnitude observed was for the EM Pivot position in pa1-debug-g at 7.72. Using the distance between the EM pivot and optical pivot as a point of comparison, the error magnitude only accounts for approximately 2%, so the error is not significant.

For verifying the calculation of the C frames, an element by element mean square error was calculated for each output. The results of this calculation can be found in Table 2. As can be observed in the table, the mean square error across all debug output files are very small, indicating the success of the program's ability to produces the C frames, with the largest error being $1.2 \cdot 10^{-5}$ for pa1-debug-e.

4.2. Testing

To test our registration algorithm, we created a point cloud of data using the `numpy.rand.random()` function. We created random translation and noise vectors the same way. The greatest magnitude of a vector in the random point cloud was on the order of 20. The magnitude of the translation was on the order of 30. The magnitude of the noise vector was on the order of 1. We then created a rotation matrix using the `scipy.stats.special_ortho_group()` function. We then transformed the generated point cloud by these known rotation, translation, and noise vectors. After to check the accuracy of our registration algorithm, we ran the algorithm on the synthetic

Table 2: Mean Square Error Between Debug Output File and Generated Output Files

Data	Mean Square Error
pa1-debug-a	$1.32 \cdot 10^{-11}$
pa1-debug-b	$2.3 \cdot 10^{-7}$
pa1-debug-c	$2.7 \cdot 10^{-8}$
pa1-debug-d	$1.4 \cdot 10^{-8}$
pa1-debug-e	$1.2 \cdot 10^{-5}$
pa1-debug-f	$7.2 \cdot 10^{-7}$
pa1-debug-g	$2.3 \cdot 10^{-6}$

data and tried to recover the known translation and rotation to within a small amount of error. We tested multiple cases to ensure our computed rotation had a determinant of 1.

To test the accuracy of the pivot calibration method, we created a vector centered at the origin with known length and lying along the positive z direction. Next, we created a random reference coordinate frame by picking a random point in the positive z half space. We then created a rotation by generating small angles in each direction (less than .5 radians) and multiplying the corresponding fundamental rotation matrices. We created 10 such rotations. We then computed the distance from the tip of the vector to our reference frame for each rotation. Finally, we fed the rotation and displacement data into the pivot calibration algorithm and checked that we got the correct length of the tip (the fixed length of the original vector) and the correct distance to the post (the distance from the reference frame to the origin). We repeated this with different tip lengths, coordinates reference frames, and rotations.

By this point, we were reasonably confident in the correctness of the registration algorithm and the pivot calibration algorithm, so it was just a matter of correctly applying them to the data. To do this, we ran the algorithms on the provided debugging data and ensured that we got the same results.

4.3. Unknown Data Results

A sample containing the first thirty lines (header, pivot positions, first C frame) of the output file for the unknown data input results can be viewed in the appendix at listings 8 to 11. Table 3 displays the results of EM pivot and optical pivot positions.

5. Discussion

From the verification of results from the given debugging data and from our own method of testing our methods, we have decent confidence in the validity of our solution. While the generated output does not match the debugging output exactly, it matches it within an acceptable margin of error in both pivot position and C frames. In defining an algorithm to compute the 3D point registration and pivot calibration, we used a closed form solution, so there is no need for an iteratvive convergence criteria. However, the approach does not use random sample consensus, and thus returned parameters are influenceable by outliers. Future versions of the generated

Table 3: EM and Optical Pivot Positions of Unknown Data

Data	Pivot Type	Position
pa1-unknown-h	EM Pivot	(202.16, 215.48, 201.71)
	Optical Pivot	(402.18, 409.18, 209.5)
pa1-unknown-i	EM Pivot	(211.43, 209.17, 195.41)
	Optical Pivot	(395.46, 393.91, 207.98)
pa1-unknown-j	EM Pivot	(192.15, 195.72, 197.1)
	Optical Pivot	(409.76, 402.98, 197.41)
pa1-unknown-k	EM Pivot	(210.55, 193.03, 191.25)
	Optical Pivot	(394.92, 396.92, 203.7)

program should attempt to account for outliers that could significantly affect returned parameters.

6. Contributions

Andrew Cornelio was responsible for a majority of the programming work and wrote the Mathematical Approach section and Testing subsection of the report. Kevin Rao was responsible for the rest of the report and wrote some preprocessing functions.

References

[1] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 698–700, 1987. DOI: 10.1109/TPAMI.1987.4767965.

[2] W. McKinney, "Data Structures for Statistical Computing in Python", in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[3] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python", *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.

[4] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del R'10, M. Wiebe, P. Peterson, P. G'erard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy", *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10.1038/s41586-020-2649-2. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2.

A. Listings

Listing 1: Program Generated pa1-debug-a-output-1.txt Sample

```
27, 8, pa1-debug-a-output-1.txt
203.49, 206.4, 198.47
390.65, 393.24, 196.47
209.93, 210.48, 211.58
208.61, 212.07, 336.56
207.29, 213.65, 461.55
210.24, 335.47, 210.0
208.92, 337.06, 334.98
207.6, 338.64, 459.97
210.55, 460.46, 208.42
209.23, 462.05, 333.4
207.91, 463.63, 458.38
334.93, 210.19, 212.9
333.61, 211.78, 337.89
332.29, 213.36, 462.87
335.23, 335.18, 211.32
333.92, 336.76, 336.31
332.6, 338.35, 461.29
335.54, 460.17, 209.74
334.22, 461.75, 334.72
332.9, 463.34, 459.71
459.92, 209.9, 214.23
458.6, 211.48, 339.21
457.28, 213.07, 464.19
460.23, 334.89, 212.65
458.91, 336.47, 337.63
457.59, 338.06, 462.61
460.54, 459.88, 211.07
459.22, 461.46, 336.05
457.9, 463.05, 461.03
```

Listing 2: Program Generated pa1-debug-b-output-1.txt Sample

```
27, 8, pa1-debug-b-output-1.txt
202.16, 190.53, 196.04
399.95, 409.91, 193.8
209.25, 209.98, 210.08
208.65, 209.45, 335.07
208.05, 208.92, 460.07
208.08, 334.97, 210.6
207.48, 334.44, 335.6
206.88, 333.91, 460.59
```

```
206.9, 459.96, 211.12
206.3, 459.43, 336.12
205.71, 458.9, 461.12
334.24, 211.15, 210.68
333.65, 210.62, 335.68
333.05, 210.09, 460.67
333.07, 336.14, 211.2
332.47, 335.61, 336.2
331.87, 335.08, 461.2
331.9, 461.14, 211.73
331.3, 460.61, 336.73
330.7, 460.08, 461.72
459.24, 212.32, 211.28
458.64, 211.79, 336.28
458.04, 211.26, 461.28
458.06, 337.31, 211.81
457.46, 336.78, 336.8
456.87, 336.25, 461.8
456.89, 462.31, 212.33
456.29, 461.78, 337.33
455.69, 461.25, 462.33
```

Listing 3: Program Generated pa1-debug-c-output-1.txt Sample

```
27, 8, pa1-debug-c-output-1.txt
191.82, 209.49, 205.01
391.23, 392.88, 192.28
210.98, 210.7, 211.78
214.39, 209.08, 336.73
217.79, 207.47, 461.67
211.08, 335.69, 213.4
214.48, 334.07, 338.34
217.88, 332.46, 463.28
211.17, 460.68, 215.01
214.57, 459.06, 339.95
217.97, 457.45, 464.9
335.94, 210.65, 208.38
339.34, 209.03, 333.32
342.74, 207.42, 458.27
336.03, 335.64, 209.99
339.43, 334.02, 334.94
342.84, 332.41, 459.88
336.12, 460.63, 211.61
339.53, 459.01, 336.55
342.93, 457.4, 461.49
460.89, 210.6, 204.98
```

```
464.29, 208.99, 329.92
467.7, 207.37, 454.86
460.98, 335.59, 206.59
464.39, 333.98, 331.53
467.79, 332.36, 456.48
461.08, 460.58, 208.2
464.48, 458.96, 333.15
467.88, 457.35, 458.09
```

Listing 4: Program Generated pa1-debug-d-output-1.txt Sample

```
27, 8, pa1-debug-d-output-1.txt
193.74, 192.93, 203.33
403.87, 394.12, 199.44
208.54, 207.5, 212.13
208.08, 206.25, 337.12
207.63, 204.99, 462.11
211.31, 332.47, 213.39
210.85, 331.21, 338.39
210.4, 329.95, 463.38
214.07, 457.43, 214.66
213.62, 456.17, 339.65
213.17, 454.91, 464.65
333.51, 204.73, 212.55
333.05, 203.47, 337.54
332.6, 202.21, 462.54
336.27, 329.69, 213.82
335.82, 328.44, 338.81
335.37, 327.18, 463.8
339.04, 454.66, 215.08
338.59, 453.4, 340.08
338.14, 452.14, 465.07
458.47, 201.96, 212.97
458.02, 200.7, 337.97
457.57, 199.44, 462.96
461.24, 326.92, 214.24
460.79, 325.66, 339.23
460.34, 324.4, 464.23
464.01, 451.88, 215.51
463.56, 450.63, 340.5
463.11, 449.37, 465.49
```

Listing 5: Program Generated pa1-debug-e-output-1.txt Sample

```
27, 8, pa1-debug-e-output-1.txt
202.4, 194.36, 204.71
409.14, 408.68, 202.79
```

```
207.71, 212.27, 211.31
204.67, 209.18, 336.23
201.63, 206.09, 461.16
208.35, 337.23, 214.41
205.3, 334.14, 339.34
202.26, 331.05, 464.26
208.98, 462.19, 217.52
205.94, 459.1, 342.44
202.9, 456.01, 467.37
332.67, 211.55, 214.33
329.63, 208.47, 339.26
326.59, 205.38, 464.18
333.31, 336.51, 217.44
330.27, 333.43, 342.37
327.22, 330.34, 467.29
333.94, 461.47, 220.55
330.9, 458.39, 345.47
327.86, 455.3, 470.4
457.63, 210.84, 217.36
454.59, 207.75, 342.29
451.55, 204.67, 467.21
458.27, 335.8, 220.47
455.23, 332.71, 345.39
452.18, 329.63, 470.32
458.91, 460.76, 223.57
455.86, 457.67, 348.5
452.82, 454.58, 473.42
```

Listing 6: Program Generated pa1-debug-f-output-1.txt Sample

```
27, 8, pa1-debug-f-output-1.txt
188.6, 200.24, 193.84
391.63, 395.37, 195.82
210.26, 210.24, 208.97
211.96, 213.77, 333.91
213.66, 217.31, 458.84
213.7, 335.14, 205.39
215.4, 338.67, 330.32
217.09, 342.21, 455.26
217.13, 460.04, 201.8
218.83, 463.58, 326.74
220.53, 467.11, 451.68
335.21, 206.76, 207.37
336.9, 210.29, 332.31
338.6, 213.83, 457.25
338.64, 331.66, 203.79
```

```
340.34, 335.19, 328.73
342.03, 338.73, 453.67
342.08, 456.56, 200.21
343.77, 460.09, 325.15
345.47, 463.63, 450.08
460.15, 203.27, 205.77
461.84, 206.81, 330.71
463.54, 210.35, 455.65
463.58, 328.17, 202.19
465.28, 331.71, 327.13
466.97, 335.25, 452.07
467.02, 453.08, 198.61
468.71, 456.61, 323.55
470.41, 460.15, 448.49
```

Listing 7: Program Generated pa1-debug-g-output-1.txt Sample

```
27, 8, pa1-debug-g-output-1.txt
185.67, 197.96, 197.93
395.48, 403.18, 201.9
210.27, 210.9, 210.06
209.77, 212.42, 335.05
209.28, 213.95, 460.04
211.61, 335.88, 208.54
211.12, 337.41, 333.53
210.63, 338.93, 458.52
212.96, 460.86, 207.02
212.47, 462.39, 332.01
211.98, 463.92, 457.0
335.26, 209.56, 210.57
334.77, 211.08, 335.56
334.27, 212.61, 460.55
336.61, 334.54, 209.05
336.11, 336.07, 334.04
335.62, 337.59, 459.03
337.95, 459.52, 207.53
337.46, 461.05, 332.52
336.97, 462.57, 457.51
460.25, 208.21, 211.07
459.76, 209.74, 336.06
459.26, 211.27, 461.05
461.6, 333.2, 209.55
461.1, 334.72, 334.54
460.61, 336.25, 459.53
462.94, 458.18, 208.03
462.45, 459.71, 333.02
```

```
461.96, 461.23, 458.01
```

Listing 8: pa1-unknown-h-output-1.txt Sample

```
27, 8, pa1-unknown-h-output-1.txt
202.16, 215.48, 201.71
402.18, 409.18, 209.5
208.56, 210.34, 211.9
209.33, 210.56, 336.89
210.1, 210.77, 461.89
209.32, 335.34, 211.68
210.09, 335.55, 336.68
210.86, 335.77, 461.67
210.09, 460.34, 211.46
210.86, 460.55, 336.46
211.62, 460.77, 461.45
333.56, 209.58, 211.13
334.32, 209.79, 336.13
335.09, 210.01, 461.12
334.32, 334.58, 210.91
335.09, 334.79, 335.91
335.85, 335.0, 460.91
335.08, 459.57, 210.69
335.85, 459.79, 335.69
336.62, 460.0, 460.69
458.55, 208.81, 210.36
459.32, 209.03, 335.36
460.09, 209.24, 460.36
459.32, 333.81, 210.15
460.08, 334.03, 335.14
460.85, 334.24, 460.14
460.08, 458.81, 209.93
460.85, 459.02, 334.93
461.61, 459.24, 459.92
```

Listing 9: pa1-unknown-i-output-1.txt Sample

```
27, 8, pa1-unknown-i-output-1.txt
211.43, 209.17, 195.41
395.46, 393.91, 207.98
209.68, 210.88, 210.67
208.28, 209.44, 335.66
206.88, 208.01, 460.64
212.22, 335.85, 212.14
210.82, 334.41, 337.12
209.42, 332.97, 462.11
214.77, 460.81, 213.6
```

```
213.37, 459.37, 338.59
211.97, 457.94, 463.57
334.65, 208.32, 212.04
333.24, 206.88, 337.03
331.84, 205.45, 462.01
337.19, 333.29, 213.51
335.79, 331.85, 338.49
334.39, 330.41, 463.48
339.73, 458.25, 214.98
338.33, 456.81, 339.96
336.93, 455.38, 464.94
459.61, 205.76, 213.42
458.21, 204.32, 338.4
456.81, 202.89, 463.38
462.16, 330.73, 214.88
460.75, 329.29, 339.87
459.35, 327.85, 464.85
464.7, 455.69, 216.35
463.3, 454.25, 341.33
461.9, 452.82, 466.32
```

Listing 10: pa1-unknown-j-output-1.txt Sample

```
27, 8, pa1-unknown-j-output-1.txt
192.15, 195.72, 197.1
405.76, 402.98, 197.41
209.15, 209.82, 211.75
207.48, 211.03, 336.73
205.81, 212.24, 461.71
211.11, 334.8, 210.56
209.44, 336.01, 335.54
207.77, 337.22, 460.53
213.07, 459.78, 209.37
211.39, 460.99, 334.36
209.72, 462.2, 459.34
334.13, 207.88, 213.44
332.46, 209.09, 338.42
330.78, 210.3, 463.4
336.08, 332.86, 212.25
334.41, 334.07, 337.23
332.74, 335.28, 462.22
338.04, 457.84, 211.06
336.37, 459.05, 336.05
334.7, 460.26, 461.03
459.1, 205.94, 215.13
457.43, 207.15, 340.11
```

```
455.76, 208.36, 465.09
461.06, 330.92, 213.94
459.38, 332.13, 338.92
457.71, 333.34, 463.91
463.01, 455.9, 212.75
461.34, 457.11, 337.74
459.67, 458.32, 462.72
```

Listing 11: pa1-unknown-k-output-1.txt Sample

```
27, 8, pa1-unknown-k-output-1.txt
210.55, 193.03, 191.25
394.92, 396.92, 203.7
208.38, 211.97, 208.37
205.12, 213.97, 333.31
201.85, 215.96, 458.26
210.9, 336.93, 206.44
207.63, 338.93, 331.39
204.37, 340.92, 456.33
213.41, 461.89, 204.52
210.15, 463.89, 329.46
206.88, 465.88, 454.4
333.32, 209.51, 211.68
330.05, 211.5, 336.62
326.79, 213.5, 461.56
335.83, 334.47, 209.75
332.57, 336.46, 334.69
329.3, 338.46, 459.63
338.34, 459.43, 207.82
335.08, 461.42, 332.76
331.81, 463.42, 457.7
458.25, 207.05, 214.98
454.98, 209.04, 339.92
451.72, 211.03, 464.86
460.76, 332.01, 213.05
457.5, 334.0, 337.99
454.23, 335.99, 462.94
463.28, 456.97, 211.12
460.01, 458.96, 336.07
456.75, 460.95, 461.01
```