EEE123
Computer Programming

Mini Project

Title:

Tabular Image Data to Matrix Data Converter
(for OpenCV purposes)

LIM WIKY 22305415 10 DEC 2022

Introduction

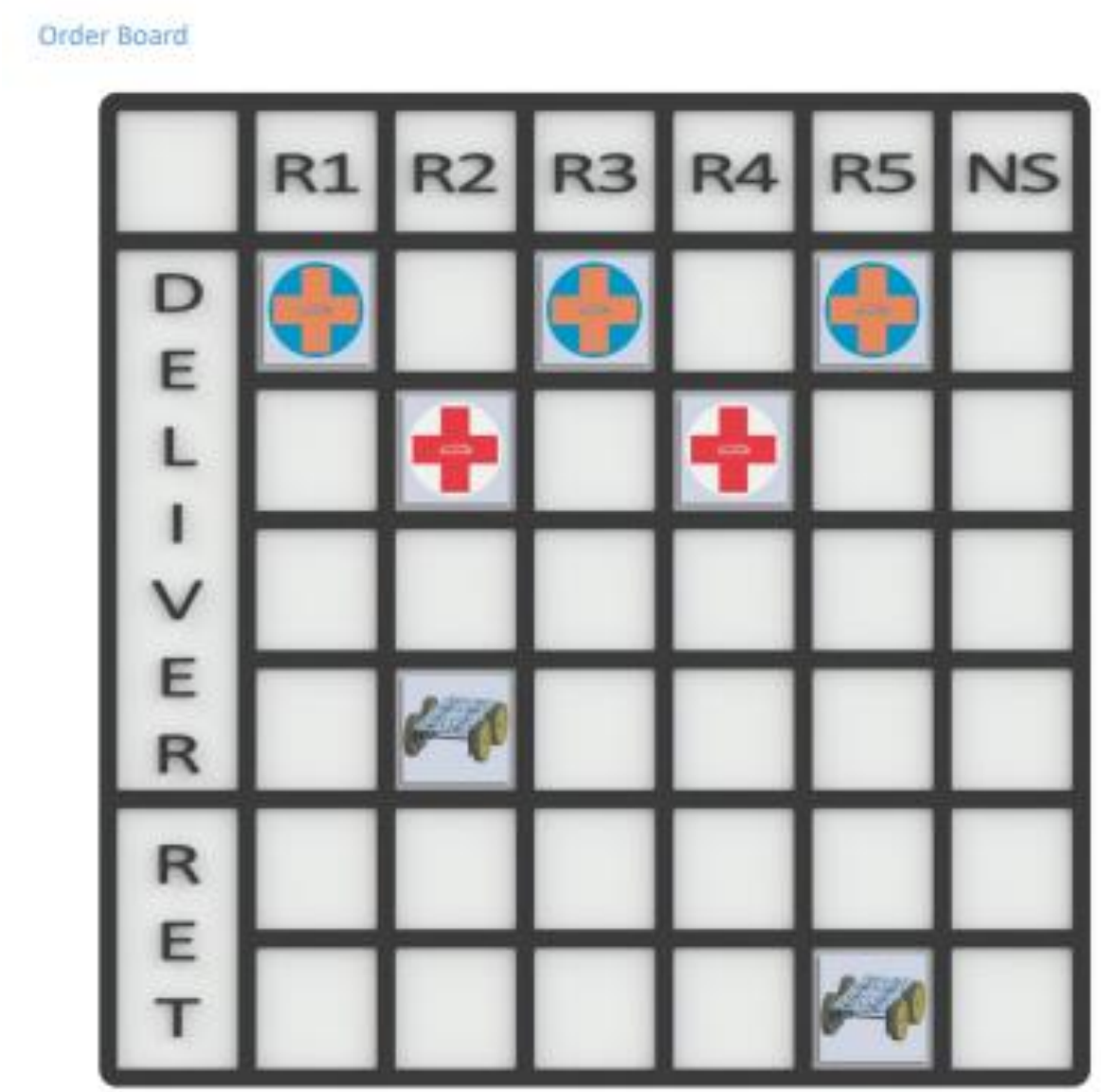
- Image to integer array conversion
- Conversion algorithm
- OpenCV application
- Learn to code

Prerequisite Knowledge

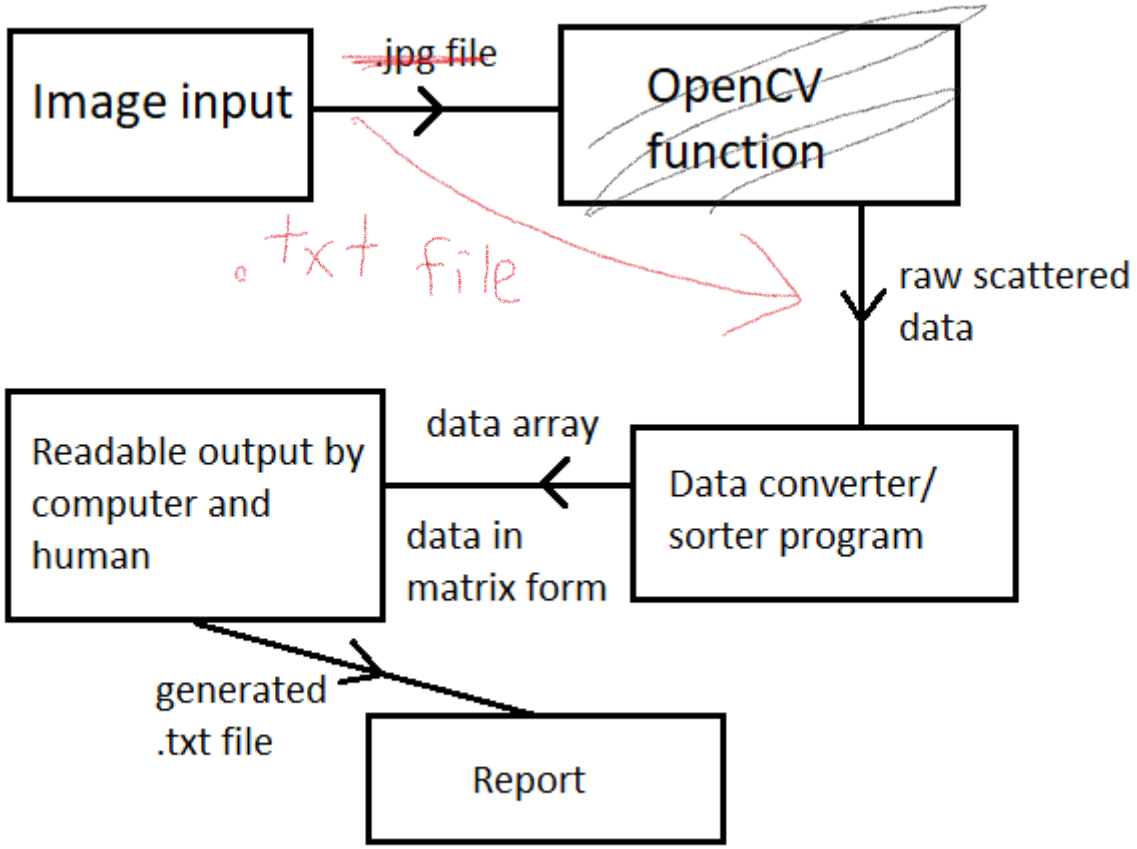
- Data structures
- Arrays
- Loops
- Functions

Problem statement

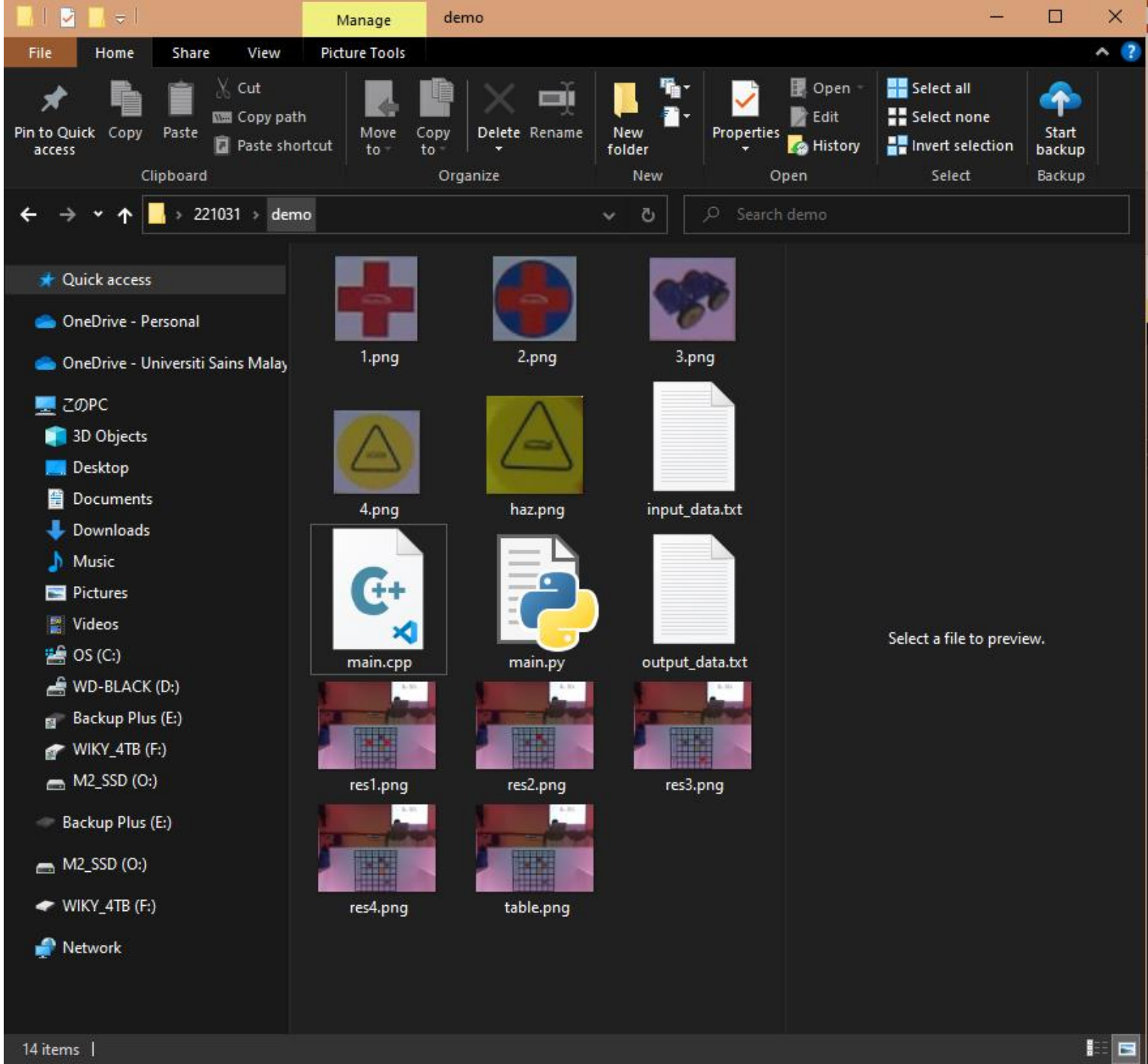
- Given an image of a table, specific image on certain cell means specific instruction.
- Generate a solution which the image in all possible combinations of configurations, can be processed to give a readable data by the computer/ program.



Program General Flow Chart



Program runs, fetch data and output data in same location



Methodology

- Get input file (image .jpg file) into program
- OpenCV process image to generate data
- Process raw data from OpenCV function (sort & convert)
- Display data in matrix form in terminal
- Output data (.txt file) in both matrix form and array form.

Methodology

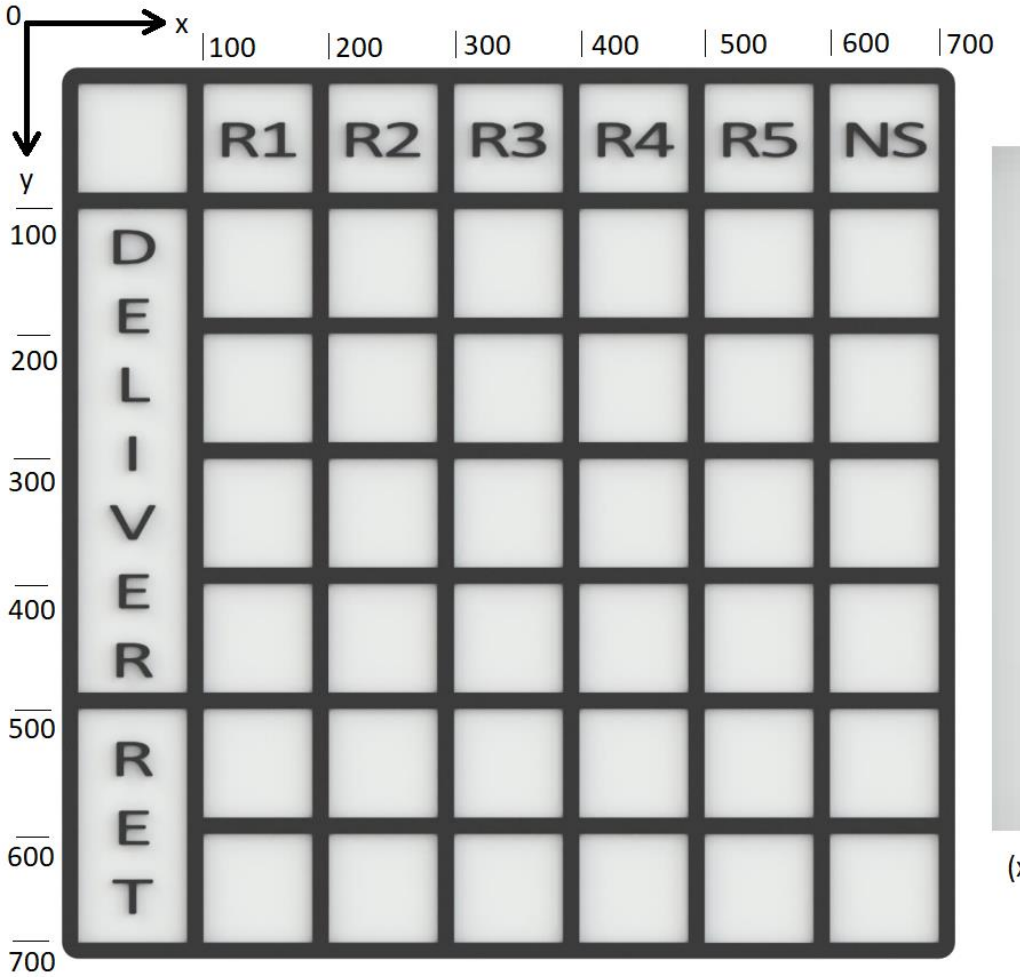
- Get input file ~~(image .jpg file)~~ (.txt file) into program
- ~~OpenCV process image to generate data~~
- Process raw data from ~~OpenCV function~~ (sort & convert)
- Display data in matrix form in terminal
- Output data (.txt file) in both matrix form and array form.

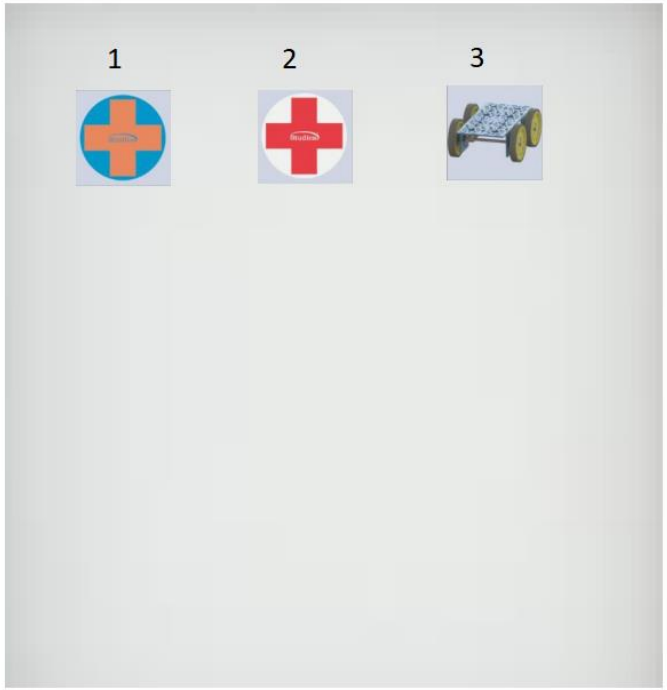
Methodology

Involves in processing coordinates

1. Raw data in (tuple)
2. Sort data ascending order
3. Convert data and append into array (2D array)
4. Add processed data into main array

Methodology

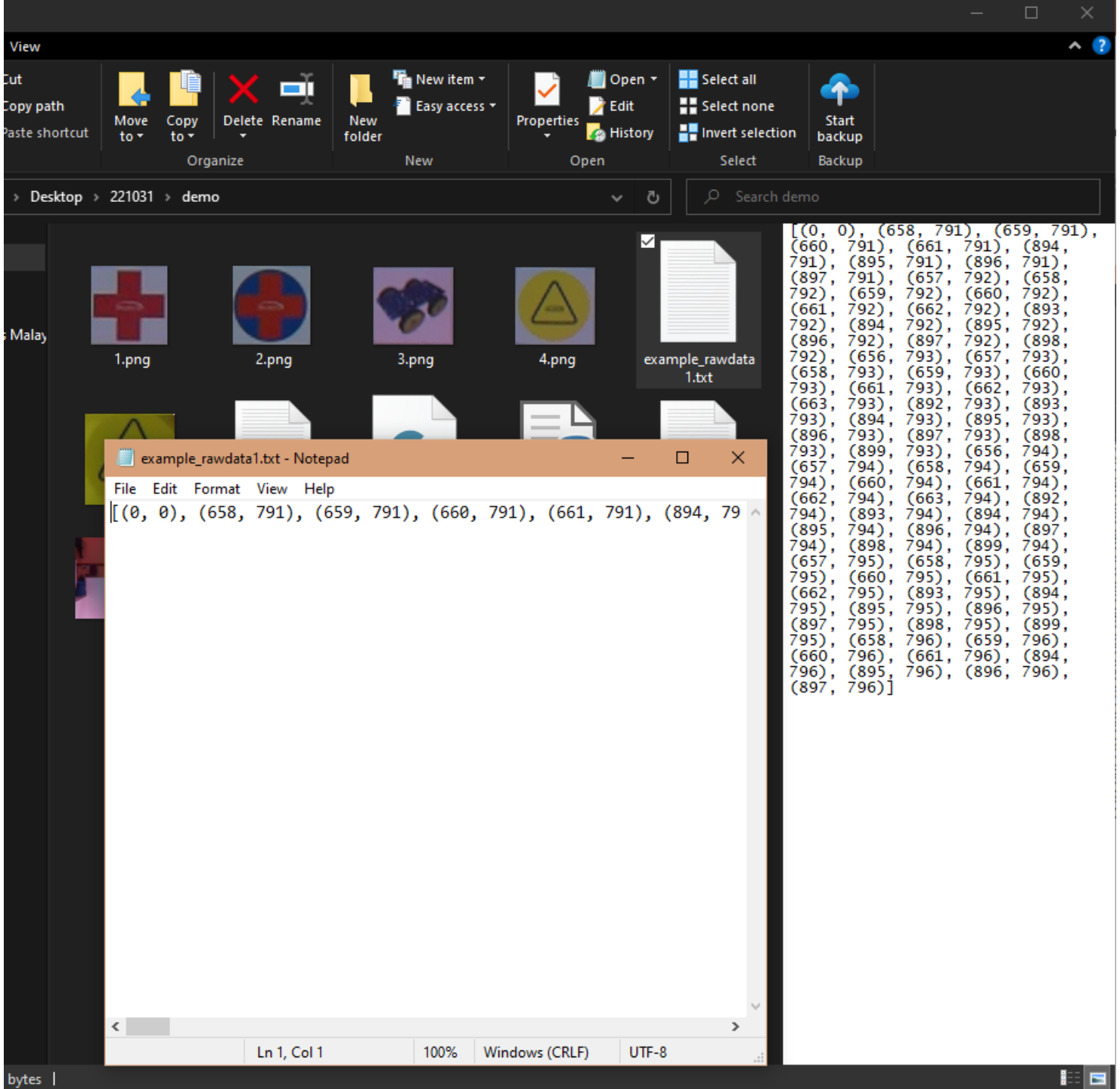




(x1, y1), (x2, y2)

coordinate null:
- (0, 0)+(100,100) to (700, 100)
- (0, 0)+(100, 100) to (100, 700)

Example of raw data



Example of raw data

unsorted

IDLE Shell 3.10.5

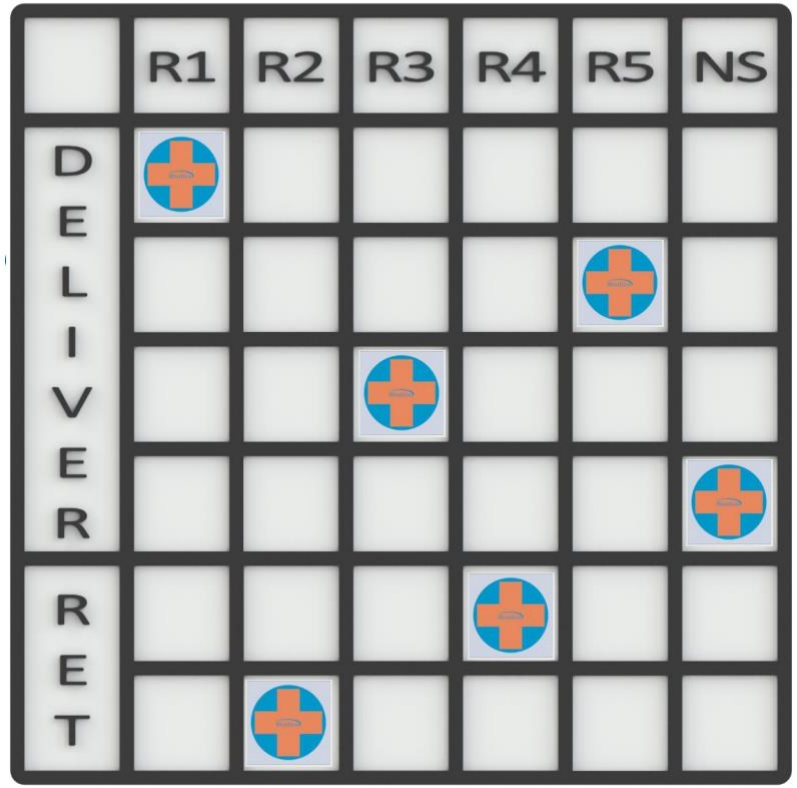
File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============== RESTART: C:\Users\User\Desktop\221031\demo\main.py ==============
[(0, 0), (658, 791), (659, 791), (660, 791), (661, 791), (894, 791), (895, 791), (896, 791), (897, 791), (6
57, 792), (658, 792), (659, 792), (660, 792), (661, 792), (662, 792), (893, 792), (894, 792), (895, 792), (
896, 792), (897, 792), (898, 792), (656, 793), (657, 793), (658, 793), (659, 793), (660, 793), (661, 793),
(662, 793), (663, 793), (892, 793), (893, 793), (894, 793), (895, 793), (896, 793), (897, 793), (898, 793),
 (899, 793), (656, 794), (657, 794), (658, 794), (659, 794), (660, 794), (661, 794), (662, 794), (663, 794)
, (892, 794), (893, 794), (894, 794), (895, 794), (896, 794), (897, 794), (898, 794), (899, 794), (657, 795
), (658, 795), (659, 795), (660, 795), (661, 795), (662, 795), (893, 795), (894, 795), (895, 795), (896, 79
5), (897, 795), (898, 795), (899, 795), (658, 796), (659, 796), (660, 796), (661, 796), (894, 796), (895, 7
96), (896, 796), (897, 796)]
[(0, 0), (814, 715), (815, 715), (816, 715), (813, 716), (814, 716), (815, 716), (816, 716), (817, 716), (8
13, 717), (814, 717), (815, 717), (816, 717), (817, 717), (818, 717), (813, 718), (814, 718), (815, 718), (
816, 718), (817, 718), (815, 719)]
[(0, 0), (892, 1024), (893, 1024), (894, 1024), (895, 1024), (896, 1024), (891, 1025), (892, 1025), (893, 1
025), (894, 1025), (895, 1025), (896, 1025), (897, 1025), (891, 1026), (892, 1026), (893, 1026), (894, 1026
), (895, 1026), (896, 1026), (897, 1026), (891, 1027), (892, 1027), (893, 1027), (894, 1027), (895, 1027),
(896, 1027), (897, 1027), (891, 1028), (892, 1028), (893, 1028), (894, 1028), (895, 1028), (896, 1028), (89
7, 1028), (891, 1029), (892, 1029), (893, 1029), (894, 1029), (895, 1029), (896, 1029), (897, 1029), (892,
1030), (893, 1030), (894, 1030), (895, 1030), (896, 1030), (894, 1031)]

[(0, 0), (656, 793), (656, 794), (657, 792), (657, 793), (657, 794), (657, 795), (658, 791), (658, 792), (6
58, 793), (658, 794), (658, 795), (658, 796), (659, 791), (659, 792), (659, 793), (659, 794), (659, 795), (
659, 796), (660, 791), (660, 792), (660, 793), (660, 794), (660, 795), (660, 796), (661, 791), (661, 792),
(661, 793), (661, 794), (661, 795), (661, 796), (662, 792), (662, 793), (662, 794), (662, 795), (663, 793),
 (663, 794), (892, 793), (892, 794), (893, 792), (893, 793), (893, 794), (893, 795), (894, 791), (894, 792)
, (894, 793), (894, 794), (894, 795), (894, 796), (895, 791), (895, 792), (895, 793), (895, 794), (895, 795
), (895, 796), (896, 791), (896, 792), (896, 793), (896, 794), (896, 795), (896, 796), (897, 791), (897, 79
2), (897, 793), (897, 794), (897, 795), (897, 796), (898, 792), (898, 793), (898, 794), (898, 795), (899, 7
93), (899, 794), (899, 795)]
[(656, 793), (892, 793)]
[(0, 0), (813, 716), (813, 717), (813, 718), (814, 715), (814, 716), (814, 717), (814, 718), (815, 715), (8
15, 716), (815, 717), (815, 718), (815, 719), (816, 715), (816, 716), (816, 717), (816, 718), (817, 716), (
817, 717), (817, 718), (818, 717)]
[(813, 716)]
[(0, 0), (891, 1025), (891, 1026), (891, 1027), (891, 1028), (891, 1029), (892, 1024), (892, 1025), (892, 1
026), (892, 1027), (892, 1028), (892, 1029), (892, 1030), (893, 1024), (893, 1025), (893, 1026), (893, 1027
), (893, 1028), (893, 1029), (893, 1030), (894, 1024), (894, 1025), (894, 1026), (894, 1027), (894, 1028),
(894, 1029), (894, 1030), (894, 1031), (895, 1024), (895, 1025), (895, 1026), (895, 1027), (895, 1028), (89
5, 1029), (895, 1030), (896, 1024), (896, 1025), (896, 1026), (896, 1027), (896, 1028), (896, 1029), (896,
1030), (897, 1025), (897, 1026), (897, 1027), (897, 1028), (897, 1029)]
[(891, 1025)]
----- m a t r i x   r e d -----
[[0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 1, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0,
 0, 0, 0, 0]]
----------------------------
```

Ln: 37 Col: 0

Scenario Example

Example 1 - One type of data (1)



Data structure : array, tuple

Required output:

	R1	R2	R3	R4	R5	NS
D	1	0	0	0	0	0
D	0	0	0	0	1	0
D	0	0	1	0	0	0
D	0	0	0	0	0	1
R	0	0	0	1	0	0
R	0	1	0	0	0	0

(Matrix form)

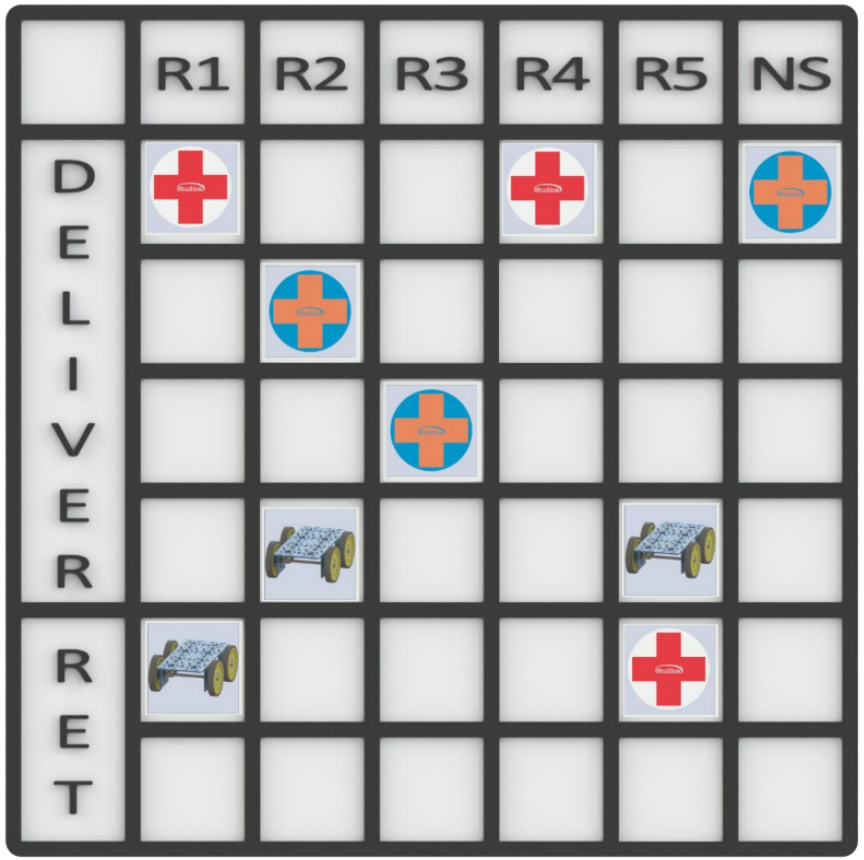
 = 1

(Array form)

data1[n] = [(100, 100), (200, 600), (300, 300), (400, 500), (500, 200), (600, 400)}

Scenario Example

Example 2 - Multiple types of data (1, 2, 3)



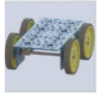
Required output:

	R1	R2	R3	R4	R5	NS
D	1	0	0	1	0	2
D	0	2	0	0	0	0
D	0	0	2	0	0	0
D	0	3	0	0	3	0
R	3	0	0	0	1	0
R	0	0	0	0	0	0

(Matrix form)

 = 1

 = 2

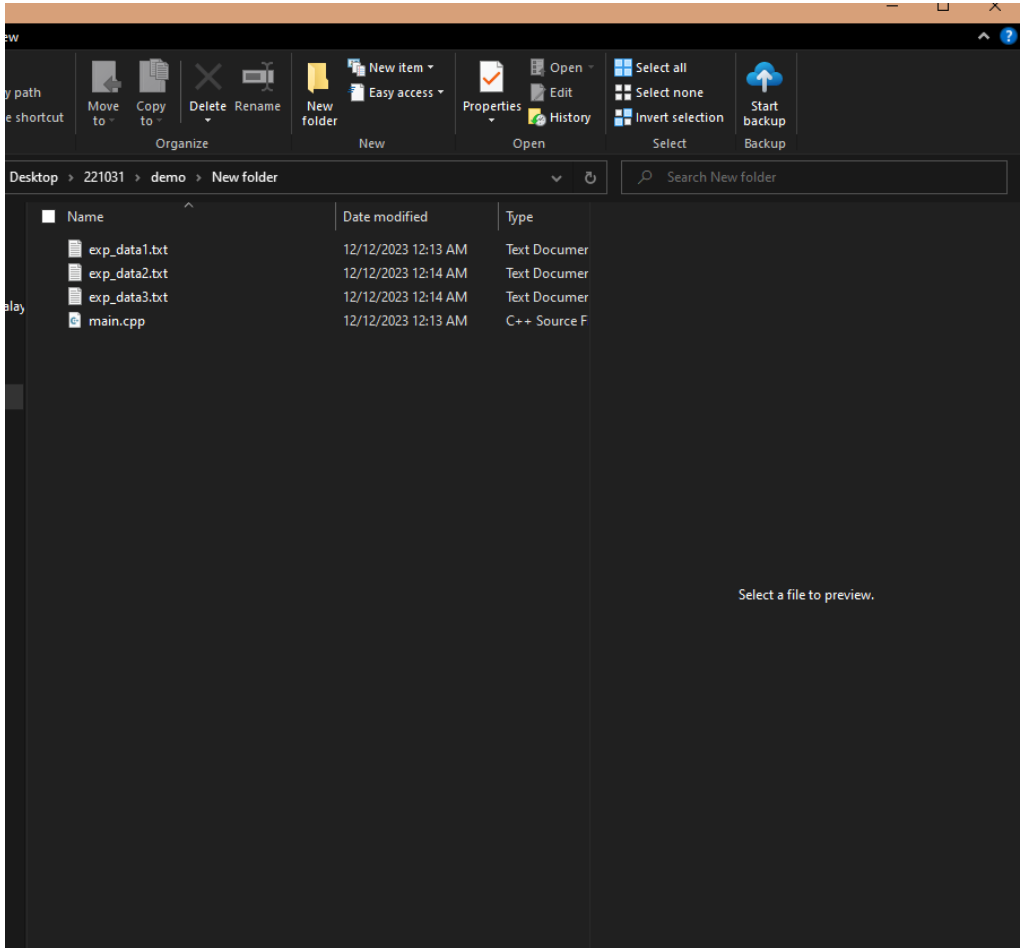
 = 3

data1[n] = [(100, 100), (400, 100), (500, 500)}
data2[n] = [(200, 200), (300, 300), (600, 100)}
data3[n] = [(100, 500), (200, 400), (500, 400)}

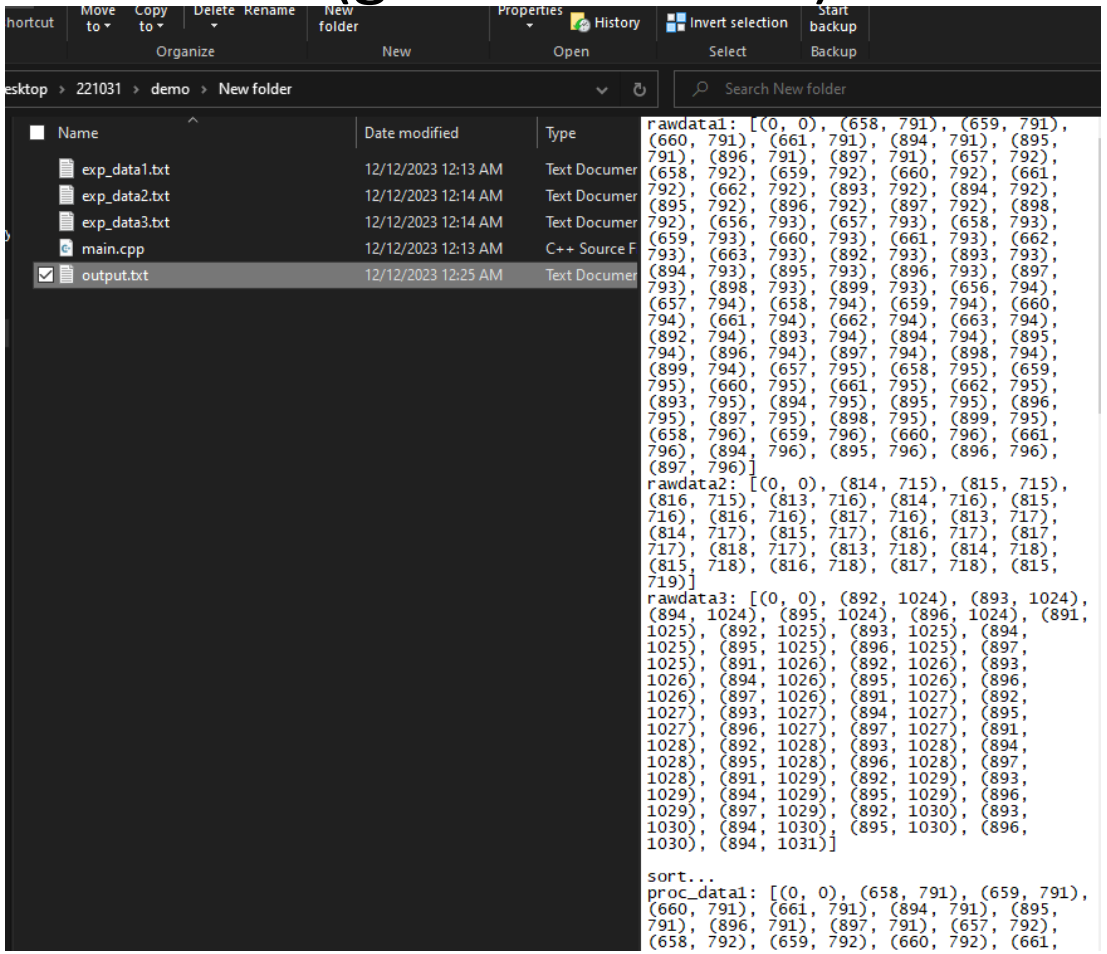
(Array form)

Goal

- Before run



- After run (generated data)



Goal

Example output data

output.txt - Notepad

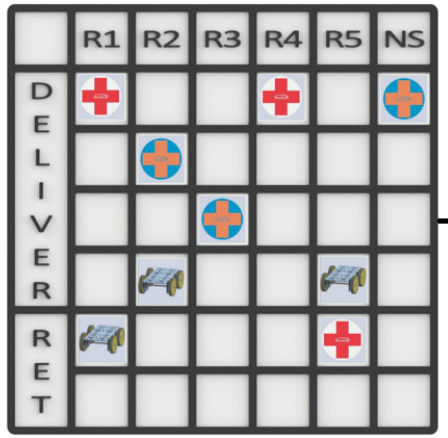
File Edit Format View Help

```
rawdata1: [(0, 0), (658, 791), (659, 791), (660, 791), (661, 791), (894, 791), (895, 791), (896, 791), (897,
rawdata2: [(0, 0), (814, 715), (815, 715), (816, 715), (813, 716), (814, 716), (815, 716), (816, 716), (817,
rawdata3: [(0, 0), (892, 1024), (893, 1024), (894, 1024), (895, 1024), (896, 1024), (891, 1025), (892, 1025)

sort...
proc_data1: [(0, 0), (658, 791), (659, 791), (660, 791), (661, 791), (894, 791), (895, 791), (896, 791), (89
proc_data2: [(0, 0), (814, 715), (815, 715), (816, 715), (813, 716), (814, 716), (815, 716), (816, 716), (81
proc_data3: [(0, 0), (892, 1024), (893, 1024), (894, 1024), (895, 1024), (896, 1024), (891, 1025), (892, 102

convert...
tempcoord1: [(0, 0), (892, 1024), (893, 1024), (894, 1024), (895, 1024), (896, 1024), (891, 1025), (892, 102
tempcoord2: [(0, 0), (814, 715), (815, 715), (816, 715), (813, 716), (814, 716), (815, 716), (816, 716), (81
tempcoord3: [(0, 0), (892, 1024), (893, 1024), (894, 1024), (895, 1024), (896, 1024), (891, 1025), (892, 102

---------------------------------- F i n a l   o u t p u t ----------------------------------------------
dataFull[][] =  {
                 {0, 0, 0, 0, 0, 0}
                 {1, 0, 2, 0, 0, 0}
                 {0, 1, 0, 3, 0, 1}
                 {0, 0, 0, 0, 2, 0}
                 {0, 2, 1, 2, 0, 0}
                 {3, 0, 1, 0, 0, 2}
                }


matrix form
_____________________________
 D   R1  R2  R3  R4  R5  NS
     0   0   0   0   0   0
     1   0   2   0   0   0
     0   1   0   3   0   1
     0   0   0   0   2   0
 R   0   2   1   2   0   0
     3   0   1   0   0   2
_____________________________

------------------------------- E n d   o f   r e p o r t ----------------------------------------------

------------------------- H a v e   a   g r e a t   d a y   : ) ---------------------------------------
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

Conclusion

input data



main.cpp

output data

```
rawdata3: [(0, 0), (892, 1024), (893, 1024), (894, 1024), (895, 1024), (896, 1024), (891, 1025), (892, 1025), (893, 1025),
(894, 1025), (895, 1025), (896, 1025), (897, 1025), (891, 1026), (892, 1026), (893, 1026), (894, 1026), (895, 1026), (896,
1026), (897, 1026), (891, 1027), (892, 1027), (893, 1027), (894, 1027), (895, 1027), (896, 1027), (897, 1027), (891, 1028),
(892, 1028), (893, 1028), (894, 1028), (895, 1028), (896, 1028), (897, 1028), (891, 1029), (892, 1029), (893, 1029), (894,
1029), (895, 1029), (896, 1029), (897, 1029), (892, 1030), (893, 1030), (894, 1030), (895, 1030), (896, 1030), (894, 1031)]

sort...
proc_data1: [(0, 0), (658, 791), (659, 791), (660, 791), (661, 791), (894, 791), (895, 791), (896, 791), (897, 791), (657,
792), (658, 792), (659, 792), (660, 792), (661, 792), (662, 792), (893, 792), (894, 792), (895, 792), (896, 792), (897, 792),
(898, 792), (656, 793), (657, 793), (658, 793), (659, 793), (660, 793), (661, 793), (662, 793), (663, 793), (892, 793), (893,
793), (894, 793), (895, 793), (896, 793), (897, 793), (898, 793), (899, 793), (656, 794), (657, 794), (658, 794), (659, 794),
(660, 794), (661, 794), (662, 794), (663, 794), (892, 794), (893, 794), (894, 794), (895, 794), (896, 794), (897, 794), (898,
794), (899, 794), (657, 795), (658, 795), (659, 795), (660, 795), (661, 795), (662, 795), (893, 795), (894, 795), (895, 795),
(896, 795), (897, 795), (898, 795), (899, 795), (658, 796), (659, 796), (660, 796), (661, 796), (894, 796), (895, 796), (896,
796), (897, 796)]
proc_data2: [(0, 0), (814, 715), (815, 715), (816, 715), (813, 716), (814, 716), (815, 716), (816, 716), (817, 716), (813,
717), (814, 717), (815, 717), (816, 717), (817, 717), (818, 717), (813, 718), (814, 718), (815, 718), (816, 718), (817, 718),
(815, 719)]
proc_data3: [(0, 0), (892, 1024), (893, 1024), (894, 1024), (895, 1024), (896, 1024), (891, 1025), (892, 1025), (893, 1025),
(894, 1025), (895, 1025), (896, 1025), (897, 1025), (891, 1026), (892, 1026), (893, 1026), (894, 1026), (895, 1026), (896,
1026), (897, 1026), (891, 1027), (892, 1027), (893, 1027), (894, 1027), (895, 1027), (896, 1027), (897, 1027), (891, 1028),
(892, 1028), (893, 1028), (894, 1028), (895, 1028), (896, 1028), (897, 1028), (891, 1029), (892, 1029), (893, 1029), (894,
1029), (895, 1029), (896, 1029), (897, 1029), (892, 1030), (893, 1030), (894, 1030), (895, 1030), (896, 1030), (894, 1031)]

convert...
tempcoord1: [(0, 0), (892, 1024), (893, 1024), (894, 1024), (895, 1024), (896, 1024), (891, 1025), (892, 1025), (893, 1025),
(894, 1025), (895, 1025), (896, 1025), (897, 1025), (891, 1026), (892, 1026), (893, 1026), (894, 1026), (895, 1026), (896,
1026), (897, 1026), (891, 1027), (892, 1027), (893, 1027), (894, 1027), (895, 1027), (896, 1027), (897, 1027), (891, 1028),
(892, 1028), (893, 1028), (894, 1028), (895, 1028), (896, 1028), (897, 1028), (891, 1029), (892, 1029), (893, 1029), (894,
1029), (895, 1029), (896, 1029), (897, 1029), (892, 1030), (893, 1030), (894, 1030), (895, 1030), (896, 1030), (894, 1031)]
tempcoord2: [(0, 0), (814, 715), (815, 715), (816, 715), (813, 716), (814, 716), (815, 716), (816, 716), (817, 716), (813,
717), (814, 717), (815, 717), (816, 717), (817, 717), (818, 717), (813, 718), (814, 718), (815, 718), (816, 718), (817, 718),
(815, 719)]
tempcoord3: [(0, 0), (892, 1024), (893, 1024), (894, 1024), (895, 1024), (896, 1024), (891, 1025), (892, 1025), (893, 1025),
(894, 1025), (895, 1025), (896, 1025), (897, 1025), (891, 1026), (892, 1026), (893, 1026), (894, 1026), (895, 1026), (896,
1026), (897, 1026), (891, 1027), (892, 1027), (893, 1027), (894, 1027), (895, 1027), (896, 1027), (897, 1027), (891, 1028),
(892, 1028), (893, 1028), (894, 1028), (895, 1028), (896, 1

---------------------------------- F i n a l   o u t p u t ----------------------------------------------
dataFull[] =        {
                 0, 0, 0, 0, 0, 0
                 1, 0, 2, 0, 0, 0
                 0, 1, 0, 3, 0, 1
                 0, 0, 0, 0, 2, 0
                 0, 2, 1, 2, 0, 0
                 3, 0, 1, 0, 0, 2
               }

matrix form
_____________________________
 D   R1  R2  R3  R4  R5  NS
     0   0   0   0   0   0
     1   0   2   0   0   0
     0   1   0   3   0   1
     0   0   0   0   2   0
 R   0   2   1   2   0   0
     3   0   1   0   0   2
_____________________________

------------------------------- E n d   o f   r e p o r t ----------------------------------------------

------------------------- H a v e   a   g r e a t   d a y   : ) ----------------------------------------
```

Job Scope

Role list

	Roles	Type	Description
1	Problem & solution seeker	Overlooker	Generate problem statement, propose method flow, prepare preliminary data
2	[Section 1] Input data	Algorithm	Propose solution and write code for inputting/sourcing data from .txt file data sets
3	[Section 2] Sorter	Algorithm	Propose solution and write code for sorting the (x, y) coordinate in array
4	[Section 3] Converter	Algorithm	Propose solution and write code for converting sorted data into matrix form
5	[Section 4] Output	Algorithm	Propose solution and write code for displaying raw data, processed data and final data in terminal (final data in matrix form), output .txt file
6	Juice mixer	Algorithm	Adjust, fix and combine all the codes (Section 1 to 4) into one main code
7	[Section 5] OpenCV (optional)	Algorithm	Write code for openCV template matching
8	Tester	Tester	Code inspection, test and validate code functionality

End of presentation

Questions?

Reference(s)

- WorldSkills 2022 Category: Mobile Robotics
- OpenCV Template Matching (Python)
- OpenCV Template Matching (C++)