---

1. You have to distribute 10 objects among 3 children. Each child has a specific value for each object given by the following table:

	A	B	C
Object 1	1	0	5
Object 2	2	1	6
Object 3	3	2	8
Object 4	4	4	7
Object 5	5	5	9
Object 6	6	3	0
Object 7	7	9	4
Object 8	8	7	3
Object 9	9	6	2
Object 10	0	8	1

   For example, **B** values object **7** at **9** units. The higher the value, the happier the child is to get the object in question.

   If a child is given some combination of objects, their *total value* for the combination is simply the sum of the values they have for the individual objects. For instance, if **C** is given the objects {3,7,8} then her total value for this "bundle" is 8 + 4 + 3 = 15.

   Suppose A is given the bundle {1,2,9,10}, B is given the bundle {3,4}, and C is given the bundle {5,6,7,8}. Then the following shows what everyone thinks of their own bundle and the bundle given to others.

	{1,2,9,10}	{3,4}	{5,6,7,8}
A	1+2+9+0 = 12	3+4 = 7	5+6+7+8 = 26
B	0+1+6+8 = 15	2+4 = 6	5+3+9+7 = 24
C	5+6+2+1 = 14	8+7 = 15	9+0+4+3 = 16

   Notice that in the situation above:

   - A values C's bundle more than their own bundle.
   - B values both A's bundle and C's bundle more than their own bundle.
   - C values their own bundle more than either A's bundle or B's bundle.

If a child values another child's bundle *more than their own,* then they are said to be jealous of the other child. In the example above, A is jealous of C and B is jealous of A and C, and C is jealous of nobody.

Consider the following process for distributing items among children:

Place all the objects on a table.

Repeat the following while there is at least one object left on the table:

1. Among the objects currently available on the table, pick the one that **A** values the most and give it to **A**.
2. Among the objects currently available on the table, pick the one that **B** values the most and give it to **B**. *If there are no objects left on the table, give nothing to B in this step.*
3. Among the objects currently available on the table, pick the one that **C** values the most and give it to **C**. *If there are no objects left on the table, give nothing to C in this step.*

Question 1A. Write down the bundles given to A,B, and C based on the algorithm above and indicate how much each of A,B,C value each of these bundles in the table below.

	Bundle given to A = {9,8,6,1}	Bundle given to B = {7,10,4}	Bundle given to C = {5,3,2}
A	**9+8+6+1 = 24**	7+0+4 = 11	5+3+2 = 10
B	7+6+3+0 = 16	**9+8+4 = 21**	5+2+1 = 9
C	2+3+0+5 = 10	4+1+7 = 12	**9+8+6 = 23**

Question 1B. What can you say about the distribution of bundles obtained at the end? Check all that apply.

(A) ✅ A will not be jealous of B
(B) ✅ A will not be jealous of C
(C) ✅ B will not be jealous of A
(D) ✅ B will not be jealous of C
(E) ✅ C will not be jealous of A
(F) ✅ C will not be jealous of B

Question 1C. What can you say about the distribution of bundles obtained at the end *for any arbitrary input*, instead of the ones given above? Check all that apply.

(A) ✅ A will not be jealous of B
(B) ✅ A will not be jealous of C

(C) ⭕ B will not be jealous of A

(D) ✅ B will not be jealous of C

(E) ⭕ C will not be jealous of A

(F) ⭕ C will not be jealous of B

> You can construct examples where B may be jealous of A and C maybe jealous of both B and A. To see this to some extent refer to the answer of the next question. However, the statements marked as true are always correct since A has the first choice in every iteration and B gets first dibs relative to C in every iteration. So, for example, you can line up A and B's bundles and compare them item by item, for each item you can observe that A will have a choice that is no worse than B's. The same is true for A v. C and B v. C.

Question 1D. Come up with an example of an input with four children **A,B,C and D** and **15 objects** for which the output has the following property:

(A) A is not jealous of B

(B) A is not jealous of C

(C) B is jealous of A

(D) B and C value each other's bundles exactly the same

(E) C is not jealous of A

(F) D is jealous of A and C but not B

Specify your input and output in the tables below.

	A	B	C	D
Object 1	110	500	1	100
Object 2	100	11	2	12
Object 3	90	10	3	11
Object 4	80	9	4	10
Object 5	1	15	10	9
Object 6	2	14	6	8
Object 7	3	13	3	7
Object 8	4	12	7	3
Object 9	5	8	50	100
Object 10	6	7	40	6
Object 11	7	6	30	5
Object 12	8	5	20	4

Object 13	9	4	5	15
Object 14	10	3	8	14
Object 15	11	2	9	13

	A's bundle = {1,2,3,4}	B's bundle = {5,6,7,8}	C's bundle = {9,10,11,12}	D's bundle = {13,14,15}
A	380	10	26	30
B	530	54	26	9
C	10	26	140	22
D	133	27	115	42

> Any valid combination works. This is a bit trickier to achieve if you restrict yourselves to values between 0 and 14 on all columns (let me know if you come up with such a solution!).

Question 1E. Generalize the algorithm above in the natural way for distributing M objects among N children. Add the N children to a queue in alphabetical order of their names (assume that all children have different names), place all objects on a table.

Let us say you have a function `find_most_valued(C,X)` which takes as input the ID of a child and some subset of X, and returns the most valued object among the objects in X.

Then the algorithm works as follows:

Repeat the following while there is at least one object left on the table:

- Let C be the child at the head of the queue.
- Let X be the current set of objects on the table.
- Let f be the output of the function `find_most_valued(C,X)`.
- Assign f to C, remove f from the table, and move C back to the tail of the queue.

How many times is the function `find_most_valued` called?

(A) ⭕ n times (B) ✅ m times (C) ⭕ nm times (D) ⭕ None of the above

> This function is called once every time the outer loop runs, which runs as many times as there are objects.

Question 1F. Suppose the queue is implemented using the `cardstack` data structure, the set of objects on the table is maintained using a linked list, and you use n arrays to track the assignments of objects to children. How long does the algorithm in the previous question take to run? Remember to account for the time taken by `find_most_valued` to do its job.

(A) ⭕ $\approx n$ (B) ⭕ $\approx nm$ (C) ⭕ $\approx m$ (D) ✅ $\approx m^2$ (E) ⭕ $\approx n^2$
(F) ⭕ None of the above

> `find_most_valued(C,X)` scans a list of at most m items to find the most valuable one. This can be done in at most m time since the input is a 2D array. The outer loop runs m times, hence the overall time is $\approx m^2$.

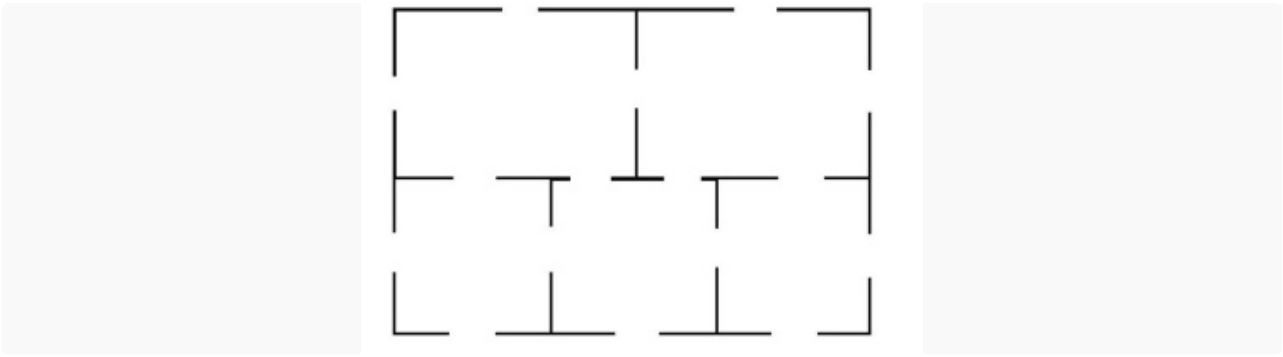
Question 1G. Suppose we have two children named A and B. Note that A is ahead of B in the queue used by the algorithm, i.e, A picks items from the table before B does. Which of the following is always true with respect to the output of the algorithm, no matter what values A and B have for the M objects? Check exactly one among the options below.

(A) ✅ A is not jealous of B
(B) ⭕ B is not jealous of A
(C) ⭕ A is not jealous of B and B is not jealous of A
(D) ⭕ None of the above

> See the answer to Question 1C for an explanation.

---

2. Consider the following 5-room apartment:



Question 2A. If we make this floor plan into a graph (keeping the next question in mind), what does it look like? Explain the association between the structure of your graph and the floor plan.

> Any well-substantiated graph based on the floor plan will work. Here is one: a vertex for every room and a special vertex for the "outer" region, with two vertices having an edge between them if the corresponding pair of locations have a door between them. Note that there is a way to draw a line that passes through each door exactly once **IFF** this graph has an Euler path (i.e, a path that visits each edge exactly once.

Question 2B. Can you find a continuous line that pass through each door exactly once? The line does not have to end where it started.

(A) ⭕ yes (B) ✅ no

Question 2C. Now we are allowed to close doors of the apartment. After closing at least how many doors we can find a continuous line that passes through each door exactly once? Answer 0 if your answer to the previous question was **YES**. Write your answer with an explanation overleaf.

> Since the graph G described in the answer to question 2A has four odd degree vertices, based on our discussions in class, we know that the answer to question 2B is no. By closing any door in any room that has an odd number of doors, we effectively remove one edge in G such that the deletion results in a graph where exactly two vertices have odd degree. This graph has an Euler path. So the answer is *one*. A different answer with a reasonably valid reasoning will also be treated as correct.