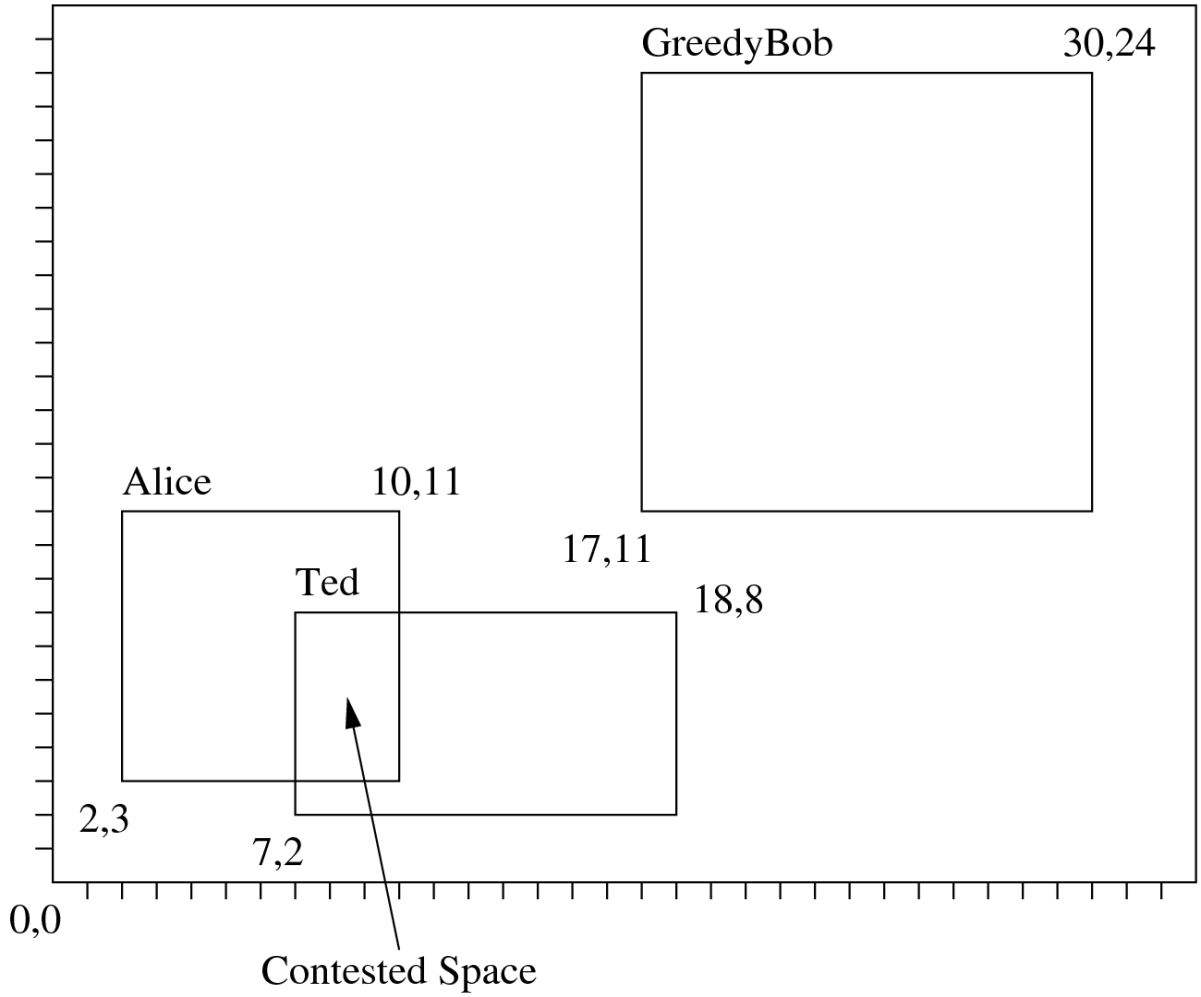
Office Space (due 21 Sep 2020)

The company that you work for is moving to a larger building. The new office is rectangular and will be subdivided into cubicles. Your employees want to request particular positions for their cubicles. You want to set up a system that lets them make their requests.

You have expressed the new office space as a coordinate system where each unit is one foot. The south-west corner of this space is assigned the coordinate (0, 0). The positive X axis is aligned with the inner edge of the building's south wall. The positive Y axis is aligned with the west wall. Employees request a position for their cubicle by giving the coordinates of the cubicle's south-west corner and the cubicle's north-east corner.



Input: As you did for the previous assignments, you will read the input from **standard input (STDIN)** in the following format:

```
33 26
3
Alice 2 3 10 11
Ted 7 2 18 8
GreedyBob 17 11 30 24
```

Each input starts with a line containing a pair of integers w and h giving the size of the new office space (w is the number of feet west-to-east, h is the number of feet south-to-north). Both of these numbers are in the range 1 to 100.

After this is a line containing an integer *n* between 1 and 20 inclusive, giving the number of employees you have. Following this are *n* cubicle placement requests, one per line.

Each request starts with the name of the employee. The name is a string of 1 to 20 characters (mix of upper and lower case letters a-z, no spaces). The name is followed by four integers x1, y1, x2, y2 where (x1, y1) indicate the coordinates of the south-west corner of the desired cubicle and (x2, y2) indicate the coordinates of the north-east corner. Each set of request coordinates satisfies $0 \le x1 \le x2 \le w$ and $0 \le y1 \le y2 \le h$.

Output: You will print your output to **standard output (STDOUT)** in the following format:

```
Total 858
Unallocated 574
Contested 15
Alice 49
Ted 51
GreedyBob 169
```

For the given input, print out a report that starts with the total number of square feet in the building and the number of square feet that no employee has requested (the unallocated space).

Next give the total number of square feet that are contested because more than one empoyee has requested the same region of the floor.

Finally, for each employee give the number of square feet that that employee can be guaranteed to have. This is the total area that they requested minus any regions that were requested by another employee.

List the employees in the same order they were given in the input.

The file (OfficeSpace.py) that you will be submitting will have the following structure. You may **NOT** change the names of the functions but you may add as many helper functions as needed. You will follow the standard coding conventions in Python.

In this problem every office space is a rectangle whose sides are oriented to x and y axis. An office space is defined by a tuple of four **integers** (x1, y1, x2, y2). Where (x1, y1) denote the lower left corner and (x2, y2) represent upper right corner.

```
# Input: a rectangle which is a tuple of 4 integers (x1, y1, x2, y2)
# Output: an integer giving the area of the rectangle
def area (rect):

# Input: two rectangles in the form of tuples of 4 integers
# Output: a tuple of 4 integers denoting the overlapping rectanlge.
#         return (0, 0, 0, 0) if there is no overlap
def overlap (rect1, rect2):

# Input: bldg is a rectangle in the form of a tuple of 4 integers
#        representing the whole office space
#        cubicles is a list of tuples of 4 integers representing
#        all the requested cubicles
# Output: a single integer denoting the area of the unallocated
#         space in the office
def unallocated_space (bldg, cubicles):

# Input: bldg is a rectangle in the form of a tuple of 4 integers
#        representing the whole office space
#        cubicles is a list of tuples of 4 integers representing
#        all the requested cubicles
# Output: a single integer denoting the area of the contested
#         space in the office
def contested_space (bldg, cubicles):

# Input: rect is a rectangle in the form of a tuple of 4 integers
#        representing the cubicle requested by an employee
#        cubicles is a list of tuples of 4 integers representing
```

```
#        all the requested cubicles
# Output: a single integer denoting the area of the uncontested
#         space in the office that the employee gets
def uncontested_space (rect, cubicles):

# Input: no input
# Output: a string denoting all test cases have passed
def test_cases ():
  assert area ((0, 0, 1, 1)) == 1
  # write your own test cases

  return "all test cases passed"

def main():
  # read the data

  # run your test cases
  '''
  print (test_cases())
  '''

  # print the following results after computation

  # compute the total office space

  # compute the total unallocated space

  # compute the total contested space

  # compute the uncontested space that each employee gets

if __name__ == "__main__":
  main()
```

You can always add more functions than those listed. You may only use standard libraries in Python.

Hint: Make use of the fact that the coordinates are all integers. Represent the office space as a 2-D list.

For this assignment you may work with a partner. Both of you must read the paper on Pair Programming and abide by the ground rules as stated in that paper. If you are working with a partner then only one of you will submit the code. Make sure that in the header in HackerRank that you have your name and UT EID and your partner's name and UT EID. If you are working alone then you will just have your name and your UT EID.

Use the *HackerRank* platform to submit your code. We should receive your work by 11 PM on Monday, 21 Sep 2020. There will be substantial penalties if you do not adhere to the guidelines. HackerRank will not assign late penalties (if any), we will make the adjustments.

- Your code must run before submission.
- You should be submitting your file through the HackerRank program. We will not accept files e-mailed to us.

This problem was taken from Kattis.