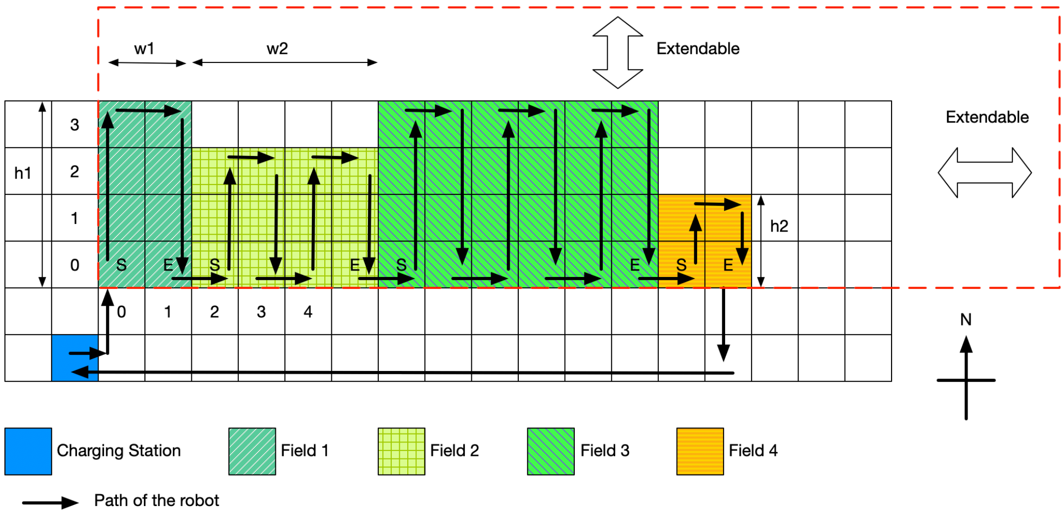
ENGG1320 Computer Programming and Applications

Assignment 1:

Due date: 4-April-2019, 23:55. Late submission: 10% discount per day.

Recently, Farmer Peter bought an irrigation robot from a startup company to water his crop. Peter has installed a charging station near his fields. Every day, the robot will leave the station and visit all Peter's fields for irrigation. Having water all the fields once, the robot will return to the station and turn into idle mode until next day (power consumption in idle mode can be ignored in this assignment)

The following diagram shows the floor plan of Peter's fields. The charging station is fixed in the blue area and the farmland is bounded by red dash lines which can extend to north and east in the diagram. The whole area is partitioned into a number of consecutive square blocks, which is the basic unit of robot movement and field area. The block (0,0) is located at the bottom left corner of the farmland. Currently, there are four fields (Field 1 to 4) that require watering. The dimension of field *i* is defined as w_i x h_i, where w_i is always an even number.



During irrigation, the robot must enter the field from the bottom left corner (marked with letter "S") and leave the field at the bottom right corner (marked with letter "E"). It waters the whole field by moving upward until the upper edge is reached, then moves one block right and moves downward until it reaches the lower edge; then the robot continues to the block on right and repeats the moving pattern until it reaches the exit point (E). When the last field has been irrigated, it will leave the exit point, and return to the charging station by taking the shortest path as shown in the diagram.

The power consumption of the robot under different activities are defined as follows:

Activities	Power consumption
moving in the non-field area (block in white)	1 unit / block

moving and watering in the field area	2 unit /block

Based on this cost model, the power consumption of one complete watering of the above fields should be 2+48*x2+ 16=114.

A. Smart Charging

At night, the robot will charge itself at the station whenever the battery level is dropped to 10% of the total capacity. However, the robot is programmed not to leave the field during irrigation, i.e. it can only visit the charging station right after finishing the irrigation of the current field. Due to this arrangement, the robot is frequently out of power in the middle of the field and Peter need to tow the robot to its station for charging.

As a computer engineer, Peter seeks for your help in modifying the AI of the robot such that it will charge itself at night if the current battery level is too low for watering all fields on the next day. Currently, Peter's fields only require being watered once per day, thus the required battery level for each day will be the power consumption of one complete watering plus 10% of battery capacity.

Input:

The first input of the program is the number (n) of fields required to water, followed by their width(w_i) and height(h_i), from left to right. The last inputs correspond to the battery capacity of the robot and the length (in term of day) of charging schedule to be generated. For a 10-days-charging schedule of a robot with 500 battery capacity that work in Peter's farm, the input should be (the text begin with # is for illustration only, they are not part of the input) :

```
4     #number of fields
2     #width of the first field, w1
4     #height of the first field, h1
4     #width of the second field, w2
3     #height of the second field, h2
6     #weight of the third field, w3
4     #height of the third field, h3
2     #width of the fourth field, w4
2     #height of the fourth field, h4
500   #battery capacity
10    #number of days to be included in the charging schedule
```

Output:

The output of the program is the day at which the robot should be charged at night. You may assume that the robot is fully charged in the morning of day 1. The output of the above input should be:

3
6

9

If the battery capacity is less than the daily requirement plus 10% power reservation, the program should output a message "Insufficient battery capacity: -x". x is the amount of missing capacity, rounded up to the nearest integer.

	Input	Output
1	4 2 4 4 3 6 4 2 2 440 20	3 6 9 12 15 18
2	4 2 4 4 3 6 4 2 2 100 20	Insufficient battery capacity: -24
3	5 3 8 2 2 5 4 4 7 6 2 500 10	2 4 6 8 10

4	1 10 4 520 30	5 10 15 20 25 30

B. Battery Saver

With your help, the robot no longer halts in the field and Peter was very happy. He invited you to have dinner together as a token of appreciation. During the dinner, you mentioned to him that lithium-ion battery, the one used in his robot, has limited life cycle. One way to preserve the battery life is to prevent largely depth of discharge (DoD), i.e. only charge the battery when the level is extremely low. You proposed reprogram the charging AI such that the robot will maintain its minimum battery level up to 50% of battery capacity. In order to achieve this goal, the robot now will visit the charging station if watering the next field will make the battery level dropped below 50% of its battery capacity. The power consumption of bringing the robot back to the charging station should also be considered. Under the new arrangement, the robot will only recharge itself at night if the power consumption of watering the first field will make the battery level less than 50%. Furthermore, as the robot may visit charging station after watering one field, the capacity requirement for the robot is now reduced to the level which allows the robot to water any one field after fully charged. It provides more flexibility to Peter in robot selection.

Input:
The input format is same as part 1.

Output:
The first line of output is the day of schedule, start with 1, following by the field number (also start with 1) and battery level, i.e. the power left after watering the field. If the robot needs to visit the charging station, output "Charging…"; If the robot needs to be charged for the next day watering at night, output "Overnight Charging…". In case the robot capacity cannot water a particular field, output
"Insufficient battery capacity to water field x: -y" where x is the field ID and y is the missing capacity.
For example, in sample input #2, power required to water field 2 alone:

Activities	Power Consumed
Transport from charging station to field 2 (row=-1, col=10)	2+10=12
Watering field 2	50*50*2=5000
Transport from field 2 (row =0, col =59) to charging station	60+2=62
Power reservation	100*0.5=50
	Total=5124

Battery capacity = 100
Missing battery capacity=5124-100= 5024

	Input	Output
1	8 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 83 1	Day 1 Field 1: completed. Battery: 73. Field 2: completed. Battery: 65. Field 3: completed. Battery: 57. Charging... Field 4: completed. Battery: 67. Field 5: completed. Battery: 59. Charging... Field 6: completed. Battery: 63. Charging... Field 7: completed. Battery: 61. Charging... Field 8: completed. Battery: 59. Overnight Charging...
2	3 10 10 50 50 30 30 100 2	Insufficient battery capacity to water field 2: -5024
3	3 6 6 1 4 8 2	Day 1 Field 1: completed. Battery: 226. Field 2: completed. Battery: 218. Field 3: completed. Battery: 186. Day 2 Field 1: completed. Battery: 95. Charging... Field 2: completed. Battery: 284. Field 3: completed. Battery: 252. Day 3 Field 1: completed. Battery: 161. Charging...

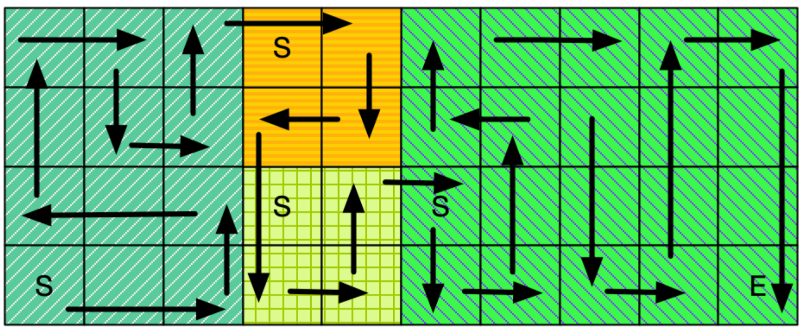
	300 3	Field 2: completed. Battery: 284. Field 3: completed. Battery: 252.

C. Smart Movement

Peter was very happy that his robot now can operate for a long time. He gave you a full basket of eggs as a token of thanks.

Before you leave, you asked him a question which bother you at the very beginning: why the width of Peter's field must be an even number while the height can be any value? Peter laughed and said, in order to make sure the robot passes every block of the fields once and moves from one field to another, the robot must return to the row 0 when it finishes watering a field. In other words, the robot starts watering at row 0 again in the next field. Having field width in even will always make the robot return to row 0 at the end.

Knowing the reason behind, you told Peter that there is another way to pass through every block once even the width is not an even number. Actually it gives a more flexible field layout like the one belows:



Peter was so impressive by your knowledge. As a farmer, Peter got many ideas to partition his field for farming. All he needs now is, given the start point always at the bottom left corner of the field, where are the possible exist points if the robot only passes all the blocks once. For the actual path, he likes to find it out by himself. Knowing what he is looking for, you open your laptop without hesitation……

Input:

Two integers, the width and height of a field.

Output:

A list of exit points in form of (row, column), one point per line. As shown in the diagram below, the bottom left corner is (0, 0), which is always the entry point of the robot. For a 2x2 field, the possible exist points are (0,1) and (1,0). Exist points should be sorted by row and then by column in ascending order.

	Input	Output
1	2 3	0,1 1,0 2,1
2	5 8	0,1 0,3 1,0 1,4 3,0 3,4 5,0 5,4 7,0 7,2 7,4
3	4 6	0,1 0,3 1,0 2,3 3,0 4,3 5,0 5,2

Submission

Virtual Programming Lab (VPL) will be setup for each part. However, we encourage you to test your program with your own test cases before submit your program. Your program should generate the output based on the specifications, i.e. without extra text/space. We may mark your program with test cases different from those found in VPL.

Plagiarism (detected by the system) submission will get zero marks, apply to the source providers as well.