# Programming Assignment 2 (Due Date: 11 Apr 2019)

This programming assignment has 5 submittables:

1. Code for Greedy Algorithms task (2%)
2. Written Answer for Greedy Algorithms task (2%)
3. Code for Dynamic Programming task (2%)
4. Code and Written Answer for Challenge Task (1%)

For the code submissions, please follow these guidelines:

- Submit in either Java or C++11 to CodeCrunch by **11 Apr 2019 2359h**, after which the portal will automatically close
- The time limit for each test case is 1 second for C++ and 5 seconds for Java for all tasks
- To get the full score for each part, you need to pass **ALL** test cases correctly and within the time limit. We may award partial marks to you after the assignment if your answer is very close to the correct answer
- Templates are provided for all the problems. You are strongly recommended to use them. **Do not change the file name or the class name of the template, or else your code will be marked as incorrect**
- Some sample test cases have been provided to you for your own testing. You can run the test script by running `java BoxStack < input_file | python check.py input_file` or `./a.out < input_file | python check.py input_file` for Java and C++ respectively. You are strongly encouraged to create your own additional test cases for your own testing.

For the written answer, please follow these guidelines:

- Write your NAME, Matric No, Tut. Gp in your Answer Sheet
- Start each problem on a separate page
- Write legibly. If we cannot read what you write, we cannot give points. In case you CANNOT write legibly, please type out your answers and print out hard copy
- To hand the homework in, staple them together and drop them into the submission slot labelled "CS3230 Assignment" at the Undergraduate Studies Office at COM1-02-19 before **11 Apr 2019 1800h**

Lastly, you should submit your own answer for the assignment. Any form of plagiarism is subject to disciplinary action. You should also not post the questions to any online forums. Any attempt to do so is also subject to disciplinary action.

# Background

Mr. Panda has started a new delivery company called Amazin which helps customers deliver packages across very long distances. The company houses all its packages inside a warehouse where they are arranged neatly so that they can be located for delivery later.

As the company grows, the number of packages it needs to manage in its warehouse has increased significantly and the human workers simply cannot keep up with the workload. Thus, the company has resorted to using automated robots to help them perform tasks around the warehouse.

There are many different tasks to be done around the warehouse and efficiency is of utmost importance in order to ensure the packages get delivered on time. Thus, Mr. Panda has hired you, his best programmer, to design and program the most efficient algorithms into the robots to perform several tasks around the warehouse.

# Greedy Algorithms (4%)

## Part 1 (2%)

A stack of $N$ packages suddenly fell down somewhere in the warehouse and you decide to send a robot to investigate the issue. The robot scans each package and finds that the packages are labelled with ID 1 to $N$. The package with ID $i$ has weight $W_i$ and is in a box that can support a total weight of $S_i$. You now want to get the robot to stack these packages back into one stack so that each package can support the total weight of all the packages above it.

### Input

The first line of input will contain an integer $N$. The next line of input contains $N$ integers representing $W_1, W_2, ..., W_N$. The next line of input contains $N$ integers representing $S_1, S_2, ..., S_N$.

### Output

Your program needs find a possible way to stack all the packages and output the IDs of the packages from the top to the bottom of the stack in one line. It is guaranteed that there is at least one way to stack the packages that satisfies the constraint. If there is more than 1 way to stack the packages, your program can output any of the ways.

### Example Input

```
4
7 3 5 4
12 7 6 14
```

### Example Output

```
3 2 4 1
```

### Explanation

The packages are stacked from top to bottom (5,6), (3,7), (4,14), (7,12). This is valid because the package (3,7) is supporting a weight of $5$, the package (4,14) is supporting a weight of $5 + 3 = 8$ and the package (7,12) is supporting a weight of $5 + 3 + 4 = 12$ which are all within the weight each package can support. Another possible solution is (3,7), (5,6), (4,14), (7,12) so outputting 2 3 4 1 will also be accepted.

An example of an invalid stack is sorting in increasing weight giving (3,7), (4,14), (5,6), (7,12) because the total weight on package (5,6) is $3 + 4 = 7$ which is more than 6. Another example of an invalid stack is sorting in increasing supporting weight giving (5,6), (3,7), (7,12), (4,14) because the total weight on package (4,14) is $5 + 3 + 7 = 15$ which is more than 14.

### Limits

- $1 \leq N \leq 2^{17}, 1 \leq W_i, S_i \leq 10^9$
- Your algorithm should have a time complexity of $O(N \log N)$

### Part 2 (2%)

Submit a written proof of the greedy algorithm you used in part 1. You should briefly describe the algorithm used in part 1 and then prove that it will always give a valid answer assuming there is a valid solution. **Take note that the assumption that there is a valid solution is important, since if there is no valid solution, no algorithm will be able to find a valid solution.**

## Dynamic Programming (3%)

An overseas shipment of $N$ packages labelled with IDs 1 to $N$ have arrived and you need to store them inside the warehouse. As all of the packages are from the same shipment, they are all packed in the same box which can support a total weight of $S$. The package with ID $i$ has a weight of $W_i$.

You want to save space by stacking up the boxes for storage, but it may not be possible to stack all the packages into one stack. Thus, you want to make a stack of packages such that the total weight of the packages in the stack is maximised. This will minimise the total weight of packages that need to be moved to other parts of the warehouse for storage.

### Input

The first line of input will contain an integer $N$ and $S$. The next line of input contains $N$ integers representing $W_1, W_2, ..., W_N$.

### Output

Your program needs find a way to stack the packages such that the total weight of the boxes in the stack is maximised, and output the IDs of the packages from the top to the bottom of the stack. If there is more than 1 way to stack the packages with the same maximum total weight, your program can output any of the ways.

### Example Input

```
7 19
10 5 30 8 5 15 5
```

### Example Output

```
1 4 3
```

### Explanation

The weights of the packages from top to bottom are $10, 8, 30$ for a total weight of $48$ and this is the maximum possible. The middle box is supporting a weight of $10$ while the bottom box is supporting a weight of $10 + 8 = 18$ both of which are less than or equal.to $19$. Another possible solution is 4 1 3 which will also be accepted.

### Limits

- $1 \leq N \times S \leq 2^{24}, 1 \leq W_i \leq 10^9$
- Your algorithm should have a time complexity of $O(N \times S)$

## Challenge Task (1%, optional)

To get the full credit for this task, you need to pass **ALL** the test cases **AND** submit a written answer containing a brief description of the algorithm and an explanation of the time complexity. Your written answer **must match** the algorithm you submitted online and we will manually mark it for the correctness. You do **NOT** need a proof of correctness but you will not receive full credit if your algorithm found to be wrong even if you passed all the test cases. However, you may receive partial credit if your algorithm passed all the test cases but has a bug not caught by the test cases.

In this task, there are also $N$ packages. The package with ID $i$ has weight $W_i$ and is in a box that can support a total weight of $S_i$, however it may not be possible to stack all the packages into one stack. Thus, you want to make a stack of packages such that the total weight of the packages in the stack is maximised.

## Input

The first line of input will contain an integer $N$. The next line of input contains $N$ integers representing $W_1, W_2, ..., W_N$. The next line of input contains $N$ integers representing $S_1, S_2, ..., S_N$.

## Output

Your program needs find a way to stack the packages such that the total weight of the boxes in the stack is maximised, and output the IDs of the packages from the top to the bottom of the stack. If there is more than 1 way to stack the packages with the same maximum total weight, your program can output any of the ways.

## Example Input

```
5
4 2 10 5 12
11 2 9 1 8
```

## Example Output

```
4 1 3
```

## Explanation

The packages are stacked from top to bottom (5,1), (4,11), (10,9). This is valid because the package (4,11) is supporting a weight of $5$, and the package (10,9) is supporting a weight of $5+4 = 9$ which are all within the weight each package can support.

## Limits

- $1 \leq N \leq 2^{18}$, $1 \leq S_i, W_i \leq 2^{11}$
- Your algorithm should have a time complexity of $O(N + S_m W_m + S_m^2 \log S_m)$ where $S_m = \max(S_i)$ and $W_m = \max(W_i)$. You need to explain this in your written answer

# Hints

## Overall

You should not need **ANY** data structures besides arrays to solve all the tasks, including the challenge task. You should also not need knowledge of any other algorithms other than Greedy Algorithms, Dynamic Programming and Sorting.

## Greedy Algorithms

The explanation of the example test case shows two ways of sorting the boxes and both do not give a valid solution for the example test case. Think about what other parameters you can sort the boxes by and under what conditions would you prefer a box to be at the top of the stack or at the bottom of the stack.

## Dynamic Programming

This task is a variant of a classic Dynamic Programming problem which has been covered in lecture. Identify the classic problem it is modified from and how to model the task to the problem.