Assignment - 1

Organizers of a workshop need your help in scheduling the presentations. There are total N presentation to be scheduled in S parallel sessions. Each session is divided uniformly in T time slots and each time slot has P presentation. For example, in Table 1 the schedule shows $S = 3$, $T = 2$, and $P = 4$, s.t. $N = T \times S \times P = 24$ presentations.

Table 1: Example

p1, p2, p3, p4	p5, p6, p7, p8
p9, p10, p11, p12	p13, p14, p15, p16
p17, p18, p19, p20	p21, p22, p23, p24

Desirable attributes of a good schedule

1. All presentations in one session should be on a single theme.
2. All presentations in parallel sessions should be on themes as far away as possible to avoid conflict.

Let us assume $d(1, 2)$ represents distance between the themes of the presentations $p1$ and $p2$, s.t. $d(1, 2) = d(2, 1)$. Further, let us assume d is in the range [0, 1]. Accordingly the similarity between the themes of the presentations $p1$ and $p2$ is measured as $u(1, 2) = 1 - d(1, 2)$.

Let us now define goodness of a schedule Sch as

$$G(Sch) = \sum_{s=1}^{s=S} \sum_{t=1}^{t=T} \sum_{\substack{C_0(i,j) \\ \text{s.t. } i,j \in s \\ i \neq j}} u(i, j) + Z \times \sum_{t=1}^{t=T} \sum_{\substack{C_1(i,j) \\ \text{s.t. } i \in s, j \notin s}} d(i, j) \tag{1}$$

where $C_0(i, j)$ represents "all pairs of presentations with same session and slot" and $C_1(i, j)$ represents "all pairs of presentations in same slot but in different (parallel) sessions". For Table 1

$$\begin{aligned} G(Sch) = u(1,2) \quad &+ \quad u(1,3) + u(1,4) + u(2,3) + u(2,4) + u(3,4) \\ &+ \quad u(5,6) + u(5,7) + ... \\ &+ \quad Z \times [d(1,9) + d(1,10) + ... + d(1,17) + d(2,18) + ... \\ &+ \quad d(5,13) + ... + d(13,21) + ...] \end{aligned}$$

The constant Z trades-off the importance of semantic coherence of one session versus reducing conflict across parallel sessions.

Our goal is to model the problem as a searching problem and find the schedule with the maximum goodness.

Input

A text file will be used as input to your program, which will contain

- Line 1: P : no. of presentations per time slot
- Line 2: S : no. of parallel sessions
- Line 3: T : no. of time slots in each session
- Line 4: Z : trade-off constant

Starting fifth line the file will have space separated list of distances between a presentation and rest others

Sample input:

```
2
2
1
1
0 0.4 0.8 1
0.4 0 0.6 0.7
0.8 0.6 0 0.3
1 0.7 0.3 0
```

Another sample input is provided along with the assignment representing similar easy problems. I recommend you experiment with other problems as well.

Output

Your program should return the max-goodness schedule.

Output format: Space separated list of paper ids, where time slots are separated by bars and sessions are separated by line.

For the above problem the optimal solution is p1 and p2 in one session; and p3 and p4 in other the other session. It will be represented as

1 2
3 4

Other equivalent ways to represent this same solution:

4 3
2 1

OR

2 1
3 4

etc. All are valid and have the total goodness of 4.4 (Verify).

Important instructions:

1. You can work in a team of maximum of 3 people.
2. You cannot use built-in libraries/implementations for search or scheduling algorithms.

3. Please do not search the Web for solutions to the problem. Your submission will be checked for **plagiarism** with the codes available on Web as well as the codes submitted by other teams. Any team found guilty will be awarded a suitable penalty as per IIT rules.

4. Your code will be automatically evaluated. You get a zero if your output is not automatically parsable.

What to submit:

Submit your code in a .zip file named in the format <RollNo>.zip. If there are two members in your team it should be called <RollNo1_RollNo2>.zip and so on. Your zip file should contain followings:

- Source code (your solution to the problem developed using Python)
- writeup.txt: 1 page write-up containing details about the search strategy you followed. Details about heuristics (if designed any) etc.

Your source code should take 2 inputs, someinputfile.txt and someoutputfile.txt. It should read the first input as input and output the answer in a file named someoutputfile.txt. for example

python xyz.py input.txt output.txt (please note any name could be given to the two files).

Python 3.5 compiler will be used during evaluation.

Evaluation:

Your submission will be auto-graded. This means that it is absolutely essential to make sure that your code follows the input/output specifications of the assignment. If you submit early you will be provided a log file which will let you know if there is any error during execution. You will have opportunity to correct the issues and resubmit. Note that the performance of your method will not be disclosed until the late submission deadline. Evaluation has two components: absolute and relative.

Absolute evaluation: Quality of the best schedule found by your method (goodness) on a test input will be the criteria. Absolute grading will be done for this component.

Relative evaluation: Relative grading among all the teams will be done for this component. Apart from quality of the best schedule found, time taken to produce the output will also be taken into account, so choose the strategy wisely.

Submission deadline: 20th Sep, 2019.

Late submission deadline and penalty: After the deadline, maximum achievable marks will be reduced by 20%. It means if you submit 5 days later to the deadline, zero marks will be awarded. This also applies to the re-submissions which are done past the Submission deadline.