1) During this semester, a lazy student attended the course of Design and Analysis of Algorithms.

In this course **k** different homeworks have been proposed by the teachers. Each homework can be either *passed*, or *failed*, or *skipped* (i.e., the student does not submit the homework). Each homework depends on a subset **H** of the **m** arguments presented during the course, with each argument contributing in equal manner at passing that homework. E.g., if the first homework is about Balanced Trees and Suffix Tries, then a student that only knows about Suffix Tries has ½ of probability of passing the homework. Note that the same argument may appear in more than one homework. The student, in order to be admitted at the final exam, must pass at least one homework.

Each argument **a** requires a given number $\mathbf{s_a}$ of hours of study to be understood. The lazy student is willing to study only at most **t** hours. The lazy student must then solve the problem of choosing which argument to study in order to maximize the probability of being admitted at the final exam.

You are required to design an algorithm that, given the **m** arguments and their associated studying hours, the **k** homeworks, each with its own set of arguments, and the thresholds **t** defined above, returns the set of arguments that the lazy student must choice in order to maximize the probability of passing the exam.

For example, suppose that there are **m=5** arguments, **A**, **B**, **C**, **D** and **E**, requiring 3, 2, 2, 5, and 4 study hour respectively. Moreover, there are three homeworks, the first about arguments **A**, **B** and **E**, the second about arguments **B**, **C**, and **D**, the third about arguments **D** and **E**. If **t=5,** then valid choices for the LazyStudent are:

- studying only one argument among **D** and **E**;
- studying two arguments among **A**, **B**, and **C** (studying only one of these arguments is clearly worse than studying two of them).

The probability of being admitted is then:

- if only argument **D** is studied, the probability of failing the first homework is 1, the probability of failing the second homework is 2/3, and the probability of failing the third homework is ½. Hence the probability of failing all of them is 1 * 2/3 * ½ = 1/3. The probability of being admitted is then 1 - 1/3 = 2/3; a similar reasoning leads the same result also if only argument **E** is studied;
- similarly, if arguments **(A, B)** or **(B, C)** are studied the probability of being admitted is 7/9;
- if arguments **(A, C)** are studied the probability of being admitted is 5/9.

Hence, the student must choose either solution **(A, B)** or solution **(B, C)**.

As another example, if **t=7**, then valid solutions[1] are

- **(D, B)**, **(D, C)**
- **(E, A)**, **(E, B)**, **(E, C)**
- **(A, B, C)**

The probability of being admitted is then as follows:

- If arguments **(D, B)** or **(E, B)** or **(A, B, C)** are studied, then the probability of being admitted is 8/9;
- If arguments **(D, C)** or **(E, A)**, or **(E, C)** are studied, then the probability of being admitted is 5/6.

Hence, the student must choose either **(D, B)** or **(E, B)** or **(A, B, C)**.

You are required then to implement a class **LazyStudent** with the following interface:

- **LazyStudent(args, hw, t)**, where **args** is a dictionary with keys being strings (denoting arguments), and values being integers (denoting study hours for that argument); **hw** is a set

1 As above, there are other valid solutions, such as choosing a single argument, or choosing two arguments among A, B, and C. However, these solution are clearly worse than the one listed here.

with each element being a tuple of strings (denoting arguments); **t** is an integer. It initializes the **LazyStudent** object.

- **choose_args()**, that must choose the arguments that the lazy student must study to maximizing the probability of being admitted at the final exam subject to the constraint of studying at most **t** hours. The function only returns the number **p** of chosen arguments.
- **next_arg()**, that returns the next chosen argument that the lazy student must study in decreasing order of required study hours. The method must have complexity **O(log p)**. The function must throw an Exception if a new request is issued after the **p-**th argument is returned.

Note that the class may have other private attributes or methods, but the public interface must coincide with the one given above.

2) Teachers must choose which students of the course of Design and Analysis of Algorithms to admit at the final exam, based on the results of their **k** homeworks.

Specifically, students are asked to report their results through the e-learning platform. The platform does not allow students to cheat, so they cannot report as passed a failed homework. However, the platform is bugged, and allows to report as skipped an exam that has been submitted and evaluated.

The teachers, based on the experience from previous years, knows the probability that a student will pass the final exam based on the results of the **k** homeworks. E.g., they know that a student passing all homeworks will pass the final exam with 95% of probability; a student passing the first homework and failing the second will pass the final exam with 60% of probability; a student passing the the first homework and skipping the second will pass the final exam with 70% of probability. Note that it may be possible that better chances of passing the final exam are achieved when an homework is skipped, than when that homework is failed.

The goal of the teachers is to maximize the probability that students admitted to the final exam will actually pass it. However, the teachers know about the bug of the e-learning platform, and thus they also are willing to disincentive that students may use this bug for increasing their probability to being admitted at the final exam.

Specifically, teachers must assign to each of the possible profiles of homework results a label (**ADM** or **NOT**) so that if a profile **h=(h$_1$, h$_2$, ...)** has **NOT** label, then no profile **h'** achieved from this by declaring as skipped one or more of the non-skipped homeworks in **h,** can have **ADM** label. Among all the assignments satisfying this property, the teachers would like to choose the one that maximizes the number of admitted students that will pass the exam, and minimizes the number of non admitted student that would pass the exam, if admitted.

For example, consider two homeworks. This implies that there are 9 results profiles: **(F, F)**, **(F, P)**, **(F, S)**, **(P, F)**, **(P, P)**, **(P, S)**, **(S, F)**, **(S, P)**, **(S, S)**, where **F** means Fail, **P** means Passed, and **S** means Skipped. From profile **(F, F)** it is possible to deviate in **(F, S)**, **(S, F)** and in **(S, S)**, thus if **NOT** label is assigned to the first profile, the same must be assigned to the remaining three profiles. Similarly, from profile **(F, P)** it is possible to deviate to profiles **(F, S)**, **(S, P)** and **(S, S)**. From profile **(F, S)** it is possible to deviate only to **(S, S)**. Similar deviations can be provided for all the remaining profiles.

Hence, the following are examples of classifications that can be returned by teachers:

- assign label **ADM** only to profile **(P, P)**;
- assign label **ADM** only to profiles **(P, P)** and **(P, F)**;
- assign label **ADM** only to profiles **(P, P)**, **(P, F)**, and **(P, S)**;
- assign label **ADM** only to profiles **(P, P)** and **(F, P)**;
- assign label **ADM** only to profiles **(P, P)**, **(F, P)**, and **(S, P)**;
- assign label **ADM** only to profiles **(P, P)**, **(P, F)**, **(F, P)**, **(P, S)**, and **(S, P)**.

Instead, for example, assigning label **ADM** only to profiles **(P, P)** and **(P, S)** is not allowed, since this will enable student to declare as skipped a failed second homework.

The probability distribution that teachers learned from previous years, states that the 9 profiles above are associated with the following probability of passing the exam: 10%, 40%, 20%, 40%, 90%, 65%, 10%, 50%, 1%. Hence,

- if we assign label **ADM** only to profile **(P, P)**, then
  - there are the 10% of students that have result profile **(F, F)** that are not admitted, but would otherwise pass the exam;

    - there are the 40% of students that have result profile **(F, P)** that are not admitted, but would otherwise pass the exam;
    - there are the 10% of students that have result profile **(P, P)** that are admitted but that will not pass the exam;

  We can do similar consideration for the remaining result profiles. In all these cases, we are essentially assigning the wrong label to the result profiles. We can then achieve a score that measures the number of these wrong labels. Specifically, we have that **score(P, P)** = 10+40+20+40+10+65+10+50+1 = 246.
- if we assign label **ADM** only to profiles **(P, P)** and **(P, F)**, then, with the same reasoning as above we have a score of 10+40+20+60+10+65+10+50+1 = 266.
- if we assign label **ADM** only to profiles **(P, P)**, **(P, F)**, and **(P, S)**, then the score is 10+40+20+60+10+35+10+50+1 = 236.
- if we assign label **ADM** only to profiles **(P, P)**, **(F, P)**, then the score is 10+60+20+40+10+65+10+50+1 = 266.
- if we assign label ADM only to profiles **(P, P)**, **(F, P)**, and **(S, P)**, then the score is 10+60+20+40+10+65+10+50+1 = 266.
- if we assign label **ADM** only to profiles **(P, P)**, **(P, F)**, **(F, P)**, **(P, S)**, and **(S, P)**, then the score is 10+60+20+60+10+35+10+50+1 = 256.

The classification that teachers would like to choose is the one with lowest score, and thus it would assign label **ADM** only to result profiles **(P, P)**, **(P, F)**, and **(P, S)**.

You are required to design and implement a function **admitted(k, distro)**, that takes in input an integer **k**, and a dictionary **distro** of $\mathbf{3^k}$ entries, whose keys are are all possible result profiles (i.e., tuples of **k** elements with each element being either **F**, or **P**, or **S**), and values are integer from 1 to 100 representing the probability that a student with that result profile will pass the final exam. The function must throw an exception if **distro** contains less that $\mathbf{3^k}$ entries. The function must return a tuple containing only result profiles to which it has been assigned the **ADM** label, so that the returned assignment has minimum score among all possible assignments.

<u>HINT</u>: The possible deviations from a result profile to another can be graphically represented with a directed graph whose nodes are result profiles and an edge from a result profile **x** to a result profile **y** exists only if **y** can be achieved from **x** by replacing some occurrence of **F** or **P** in **S**. Essentially we are asking that the endpoint of these edges are never separated.

Similarly, the contribution of each profile to the score can be graphically represented having node representing result profile **x** to be linked to two special nodes **ADM** and **NOT**, with weights **distro[x]** and **100-distro[x]**, respectively. The meaning is that **x** contributes to the score **distro[x]** if the corresponding node is separated from **ADM** and **100-distro[x]** if it is separated from **NOT**.