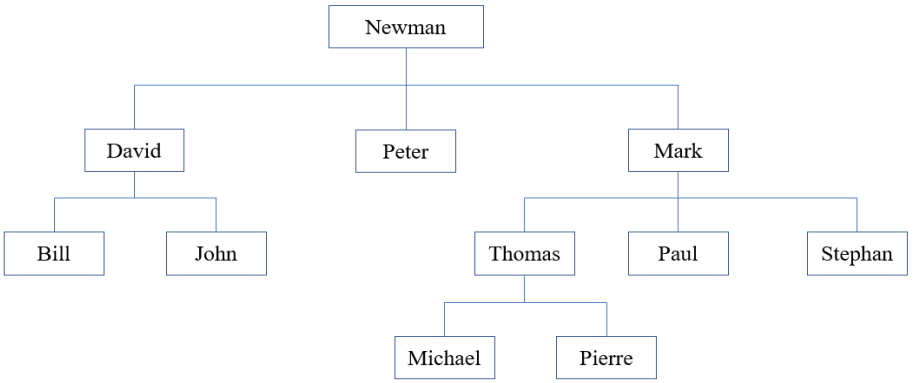
MINI PROJECT DESCRIPTION

1. FAMILY

Information (parent-childrent relation) about a family is stored in an external TEXT file:

- Each line contains two strings (separated by a white space) which are the names of a child and a parent.



family generation tree

```
Peter  Newman
Michael  Thomas
John  David
Paul  Mark
Stephan  Mark
Pierre  Thomas
Mark  Newman
Bill  David
David  Newman
Thomas  Mark
```

Text file storing family generation information

Write a program (running into an interactive mode) to perform following operations

- Load data from the text file into the memory and build a generation tree and print an error message if the data is not consistent (for example, no unique highest ancestor; there is a cycle: A is a parent of B, B is a parent of C, and C is a parent of A)
- User inputs a name of a person and the program prints out the level of generation of the given person (example, if user input “Thomas”, the program must output: “3th generation”)
- User inputs a name of a person, the program prints out the number of descendants of the given person

2. SEARCH TEXT

Given a text T and a pattern P which are represented by sequences of characters taken from the set {a,…,z,A,…,Z,0,1,2,3,4,5,6,7,8,9} and the BLANK character. Find all the positions that P appears on T.

Input

- Line 1 contains P (length of P is less than or equals to 10^5)
- Line 2 contains T (length of T is less than or equals to 10^6)

Output

Number of positions that P appears on T.

Example

Input	**Output**
computer There are so many kinds of computers today including high performance computers, laptop computers. Mobile phones are also computers	4

METHOD HINT: Use Hashing approach

- Convert the given pattern P (length k) into an integer (hash code)

$h(P) = (P[0]*256^{k-1} + P[1]*256^{k-2} + \ldots + P[k-1]*256^0)$ modulo Q (for a given constant which is a positive integer Q)

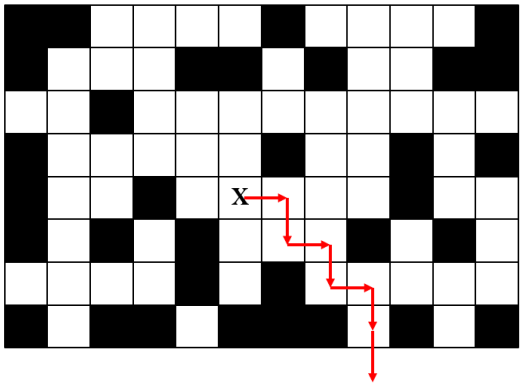
- Convert subsequence of T (for each position when P shifts from left to right on T) into an integer (hash code)
- Compare the two hash codes, if these codes are equal, then compare the two string character by character.

3. MAZE

A Maze is represented by a 0-1 matrix a_{NxM} in which $a_{i,j} = 1$ means cell (i,j) is an obstacle, $a_{i,j} = 0$ means cell (i,j) is free.

From a free cell, we can go up, down, left, or right to an adjacent free cell.

Compute the minimal number of steps to escape from a Maze from a given start cell (i_0, j_0) within the Maze.



Escape the Maze after 7 steps

- **Input**
  - Line 1 contains N, M, i_0, j_0 ($2 \le N,M \le 900$)
  - Line i+1 (i = 1,…, N) contains the ith line of the matrix $a_{N\text{x}M}$
- **Output**
  - Unique line contains the number minimal of steps to escape the Maze or -1 if no way to escape the Maze.
- **Example**

Input	**Output**
8 12 5 6 1 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 1 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 1 0 1 1 1 0 1 0 1	7

4. **Bus scheduling**

There are n passengers 1, 2, …, n. The passenger i want to travel from point i to point $i + n$ (i = 1,2,…,n). There is a bus located at point 0 and has k places for transporting the passengers (it means at any time, there are at most k

passengers on the bus). You are given the distance matrix c in which $c(i,j)$ is the traveling distance from point i to point j ($i, j = 0,1,\ldots, 2n$). Compute the shortest route for the bus, serving n passengers and coming back to point 0 (the route visits each point 1, 2, …, 2n exactly once).

- **Input**
  - Line 1 contains n and k ($1 \le n \le 11$, $1 \le k \le 10$)
  - Line i+1 (i = 1, 2,…, 2n+1) contains the (i-1)th line of the matrix c (rows and columns are indexed from 0,1,2,..,2n)
- **Output:**
  - Unique line contains the length of the shortest route
- **Example**

Input	Output
3 2 0 8 5 1 10 5 9 9 0 5 6 6 2 8 2 2 0 3 8 7 2 5 3 4 0 3 2 7 9 6 8 7 0 9 10 3 8 10 6 5 0 2 3 4 4 5 2 2 0	25

Explanation

There are 3 passengers need to be transported. The bus has 2 places for passengers (at any moment, there are at most 2 passengers on the bus)

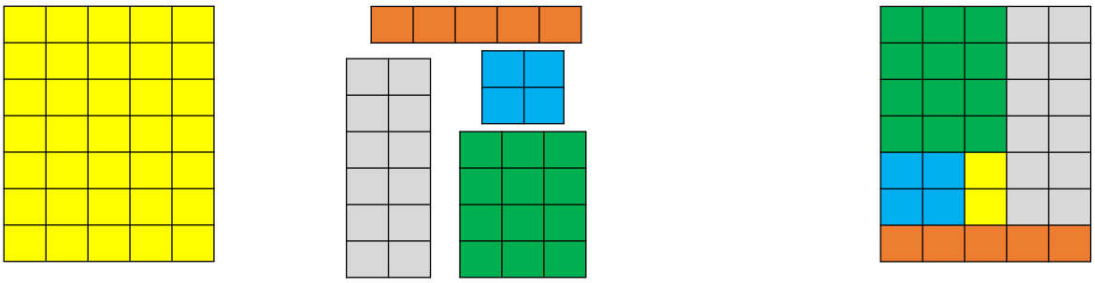
The optimal route of the bus is: 0 → 3 → 1 → 4 → 2 → 6 → 5 → 0, and the length of the optimal route is c(0,3) + c(3,1) + c(1,4) + c(4,2) + c(2,6) + c(6,5) = 1 + 3 + 6 + 8 + 2 + 2 + 3 = 25

That means: the bus goes to pickup passenger 3 and then pickup passenger 1, then goes to delivery passenger 1, then goes to pickup passenger 2, then goes to

delivery passenger 3, then goes to delivery passenger 2, and finally comes back to the depot 0.

5. Material Cut

Given a material having the shape rectangle with height H and width W. We need to cut this material into n smaller rectangle submaterials of size (h_1,w_1), (h_2, w_2), ..., (h_n, w_n). Write a program to check if we can perform this cut.



Material | Sub Materials need to be cut | Cut Solution

Input

- Line 1: contains two positive integers H, W (1 <= H, W <= 10)
- Line 2: contains a positive integer n (1 <= n <= 10)
- Line i+2 (i= 1,...,n): contains two positive integer h_i and w_i (1 <= h_i, w_i <= 10)

Output

Write 1 if we can perform the cut and write 0, otherwise.

Example

Input	Output
7 5 4 1 5 6 2 2 2 4 3	1

6. Balanced Academic Curriculum Problem

The BACP is to design a balanced academic curriculum by assigning periods to courses in a way that the academic load of each period is balanced .

There are N courses 1, 2, …, N that must be assigned to M periods 1, 2, …, M. Each course i has credit c_i and has some courses as prerequisites. The load of a period is defined to be the sum of credits of courses assigned to that period. The prerequisites information is represented by a matrix A_{NxN} in which $A_{i,j} = 1$ indicates that course i must be assigned to a period before the period to which the course j is assigned. Compute the solution satisfying constraints

- Satisfy the prerequisites constraints: if $A_{i,j} = 1$, then course i must be assigned to a period before the period to which the course j is assigned
- The maximum load for all periods is minimal

- **Input**
  - Line 1 contains N and M ($2 \leq N \leq 16$, $2 \leq M \leq 5$)
  - Line 2 contains $c_1, c_2, \ldots, c_N$
  - Line i+2 (i = 1,…, N) contains the i^{th} line of the matrix A
- **Output**
  - Unique line contains that maximum load for all periods of the solution found
- **Example**

Input	Output
6 2 4 4 4 4 2 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0	12

7. **Paper Reviewers Assignment Problem**

The chair of a conference must assign scientific papers to reviewers in a balance way. There are N papers 1, 2, …, N and M reviewers 1, 2, …, M. Each paper i has a list $L(i)$ of reviewers who are willing to review that paper.

A review plan is an assignment reviewers to papers. The load of a reviewer is the number of papers he/she have to review.
Given a constant b, compute the assignment such that

- Each paper is reviewed by exactly b reviewers
- The maximum load of all reviewers is minimal

- **Input**
  - Line 1 contains N, M and b
  - Line i+1 (i = 1,…,N) contains a positive integer k followed by k positive integers representing the list L(i)
- **Output**
  - Unique line contains the maximum load for all reviewers of the solution found or contains -1 if no solution found.
- **Example**

Input	Output
10 6 2 3 6 5 4 5 4 6 3 1 2 5 6 3 2 4 1 4 1 6 2 3 5 4 5 2 6 3 2 4 6 4 1 5 4 2 2 5 3 3 2 3 1 5 5 6 3 2 1	4

8. **Balanced Course Assignment**
   a. **Problem description**

At the beginning of the semester, the head of a computer science department D have to assign courses to teachers in a balanced way. The department D has m teachers T={1, 2, ..., m} and n courses C={1, 2,..., n}. Each teacher $t \in T$ has a preference list which is a list of courses he/she can teach depending on his/her specialization. We know a list of pairs of conflicting two courses that cannot be assigned to the same teacher as these courses have been already scheduled in the same slot of the timetable. The load of a teacher is the number of courses assigned to her/him. How to assign n courses to m teacher such that each course assigned to a teacher is in his/her preference list, no two conflicting courses are assigned to the same teacher, and the difference between maximal load and minimal load is minimal.

InputFile The input consists of following lines

- Line 1: contains two integers m and n ($1 \leq m \leq 10$, $1 \leq n \leq 30$)
- Line i+1: contains an positive integer k and k positive integers indicating the courses that teacher i can teach ($\forall\ i = 1,\ldots, m$)
- Line m+2: contains an integer k
- Line i+m+2: contains two integer i and j indicating two conflicting courses ($\forall\ i$=1,. . . , k)

OutputFile

The output contains a unique number which is the maximal load of the teachers in the solution found and the value -1 if not solution found.

Example:

Input	Ouput
4 9 **4 1 2 3 5** **5 1 2 5 6 8** **6 1 2 4 5 7 9** **4 3 4 5 9** **13** **1 3** **1 2** **2 5** **1 8** **1 4** **4 9** **5 7** **2 3** **1 7** **6 9** **4 8** **3 5**	**3**

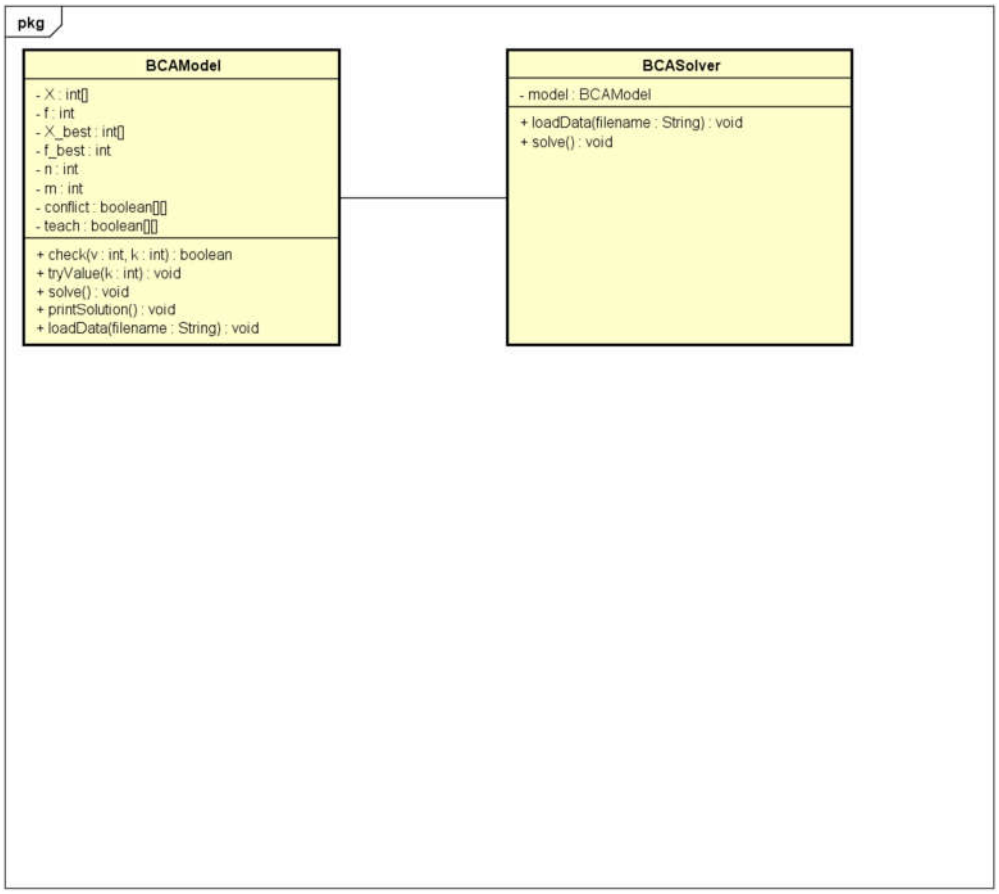
2 7	

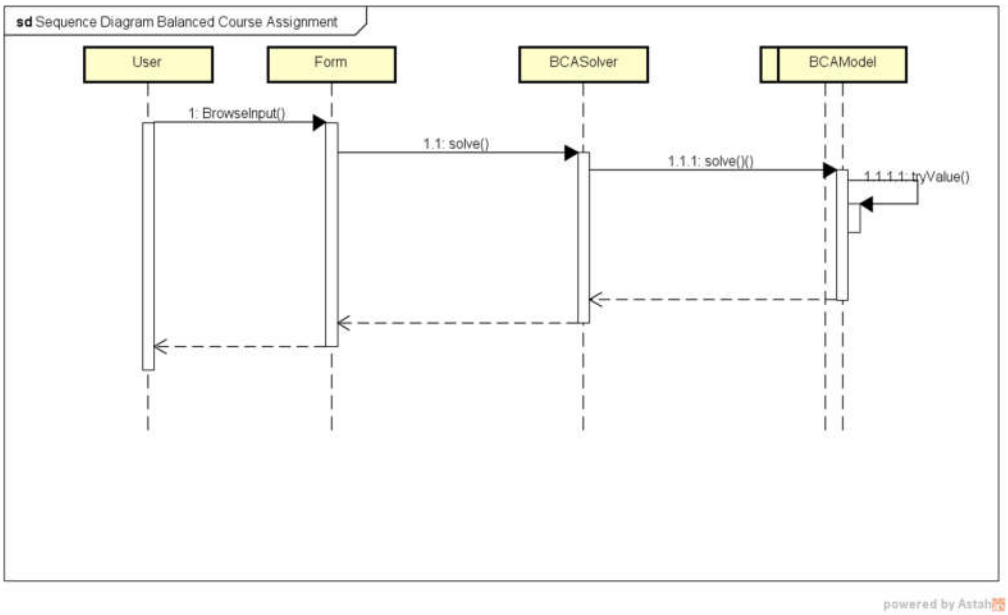
Courses assigned to teacher 1: 1, 5
Courses assigned to teacher 2: 2, 6, 8
Courses assigned to teacher 3: 4, 7
Courses assigned to teacher 1: 3, 9

Maximal load = 3, minimal load = 2

b. Design

- Use backtracking search to explore all possibilities in the search space.
- Decision variables X[] in which X[i] represents the index of teacher assigned to the course i.
- Method tryValue(int k) tries all values for X[k]
- Method check(int v, int k) returns true if value v can be assigned to X[k] without violating constraints of the problem
- Data structures
    - conflict[i][j] = true if courses i and j conflict to each other
    - teach[t][i] = true if teacher t can teach course i





9. Sudoku

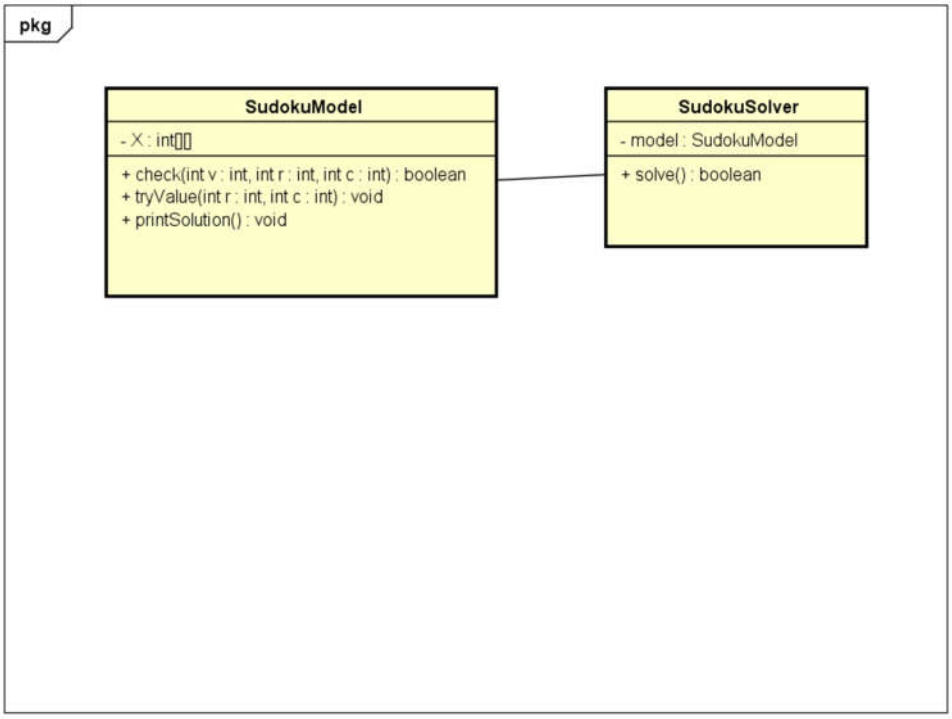
a. Problem description

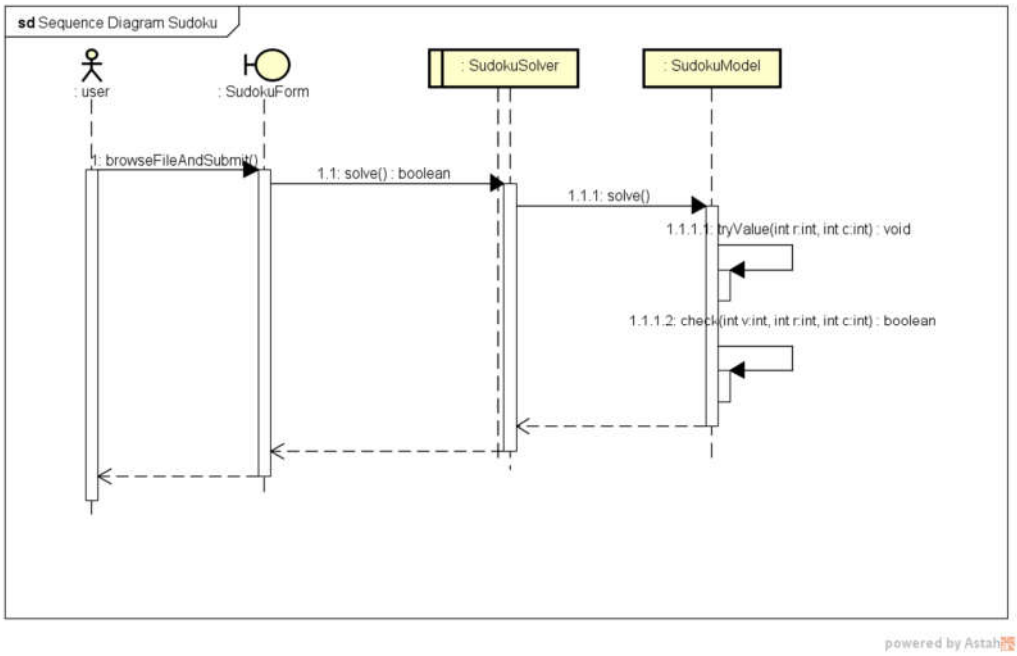
There is an incomplete 9x9 table of numbers. Fill each empty cell a number from 1 to 9 satisfying following constraints

- Numbers within any of the 9 individual 3x3 boxes are different
- Numbers within any row of the given 9x9 table are different
- Numbers within any column of the given 9x9 table are different

b. Design

- Use backtracking search to explore all possibilities in the search space.
- Decision variables X[][] in which X[i][j] represents the value assigned to the cell row i and column j of the table.
- method tryValue(int i, int j) tries all values for cell (i, j) of the table
- method check(int v, int i, int j) returns **true** if v can be assigned to cell (i, j) without violating constraints of the problem.





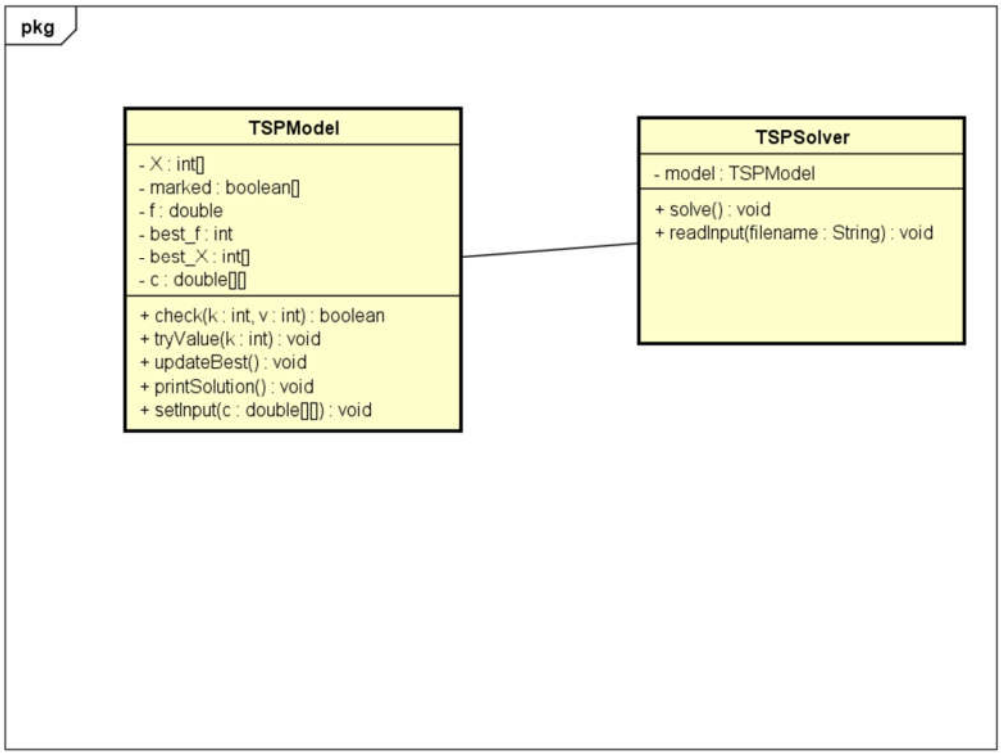
10. Shiping route

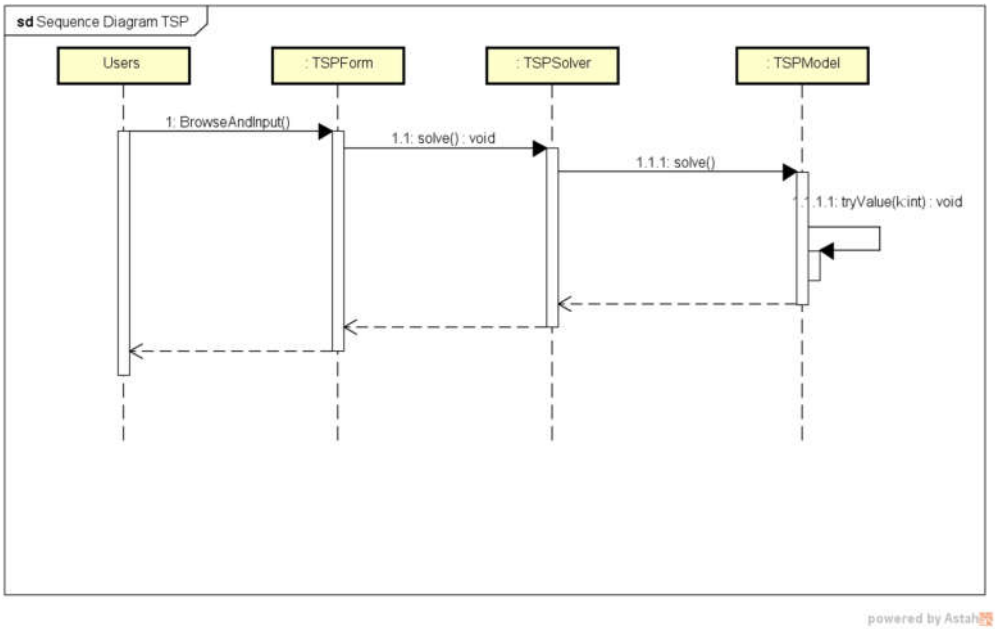
a. Problem description

A shipper must delivery goods to N customers represented by points 1, 2, …, N from the store represented by the point 0. The travel distance from point i to point j is $c_{i,j}$ ($i, j = 0,1,\ldots,N$). Find the shipping route from the store, visiting N customers and returning to the store such that to total travel distance is minimal.

b. Design

- Use backtracking search to investigate all candidate solutions in the solution space.
- Decision variables X[] in which X[i] is the i^{th} point of the route, for all i = 0, 1, 2, …, N: the route of the shipper is X[0] → X[1] → X[2] → . . . → X[N] → X[0] (X[0] = 0 is the starting point).
- method tryValue(int k) tries values for X[k]
- method check(int v, int k) returns true if value v can be assigned to X[k] without violating constraints of the problem (each customer point is visited exactly once).





InputFile The input consists of following lines

- Line 1: contains an integer N ($1 \leq N \leq 20$)
- Line i+1 (i=1,2,…,N+1): contains the (i-1)th row of matrix c

OutputFile
The output contains a unique number which is the maximal load of the teachers in the solution found and the value -1 if not solution found.

Example:

Input	Ouput
3 **0 4 1 8** **1 0 7 5** **7 6 0 1** **8 2 6 0**	**5**

Shortest route is: 0 – 2 – 3 – 1 – 0 and the distance is $c_{0,2} + c_{2,3} + c_{3,1} + c_{1,0} = 1 + 1 + 2 + 1 = 5$

11. Water Jug

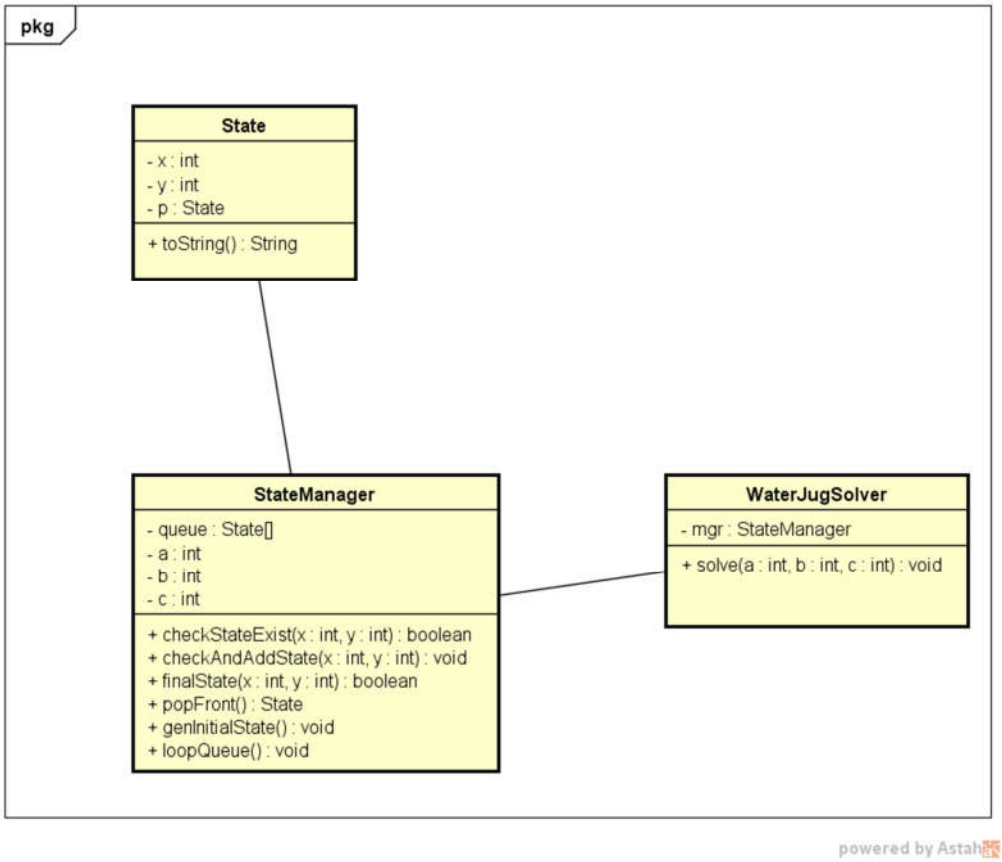
a. Problem description

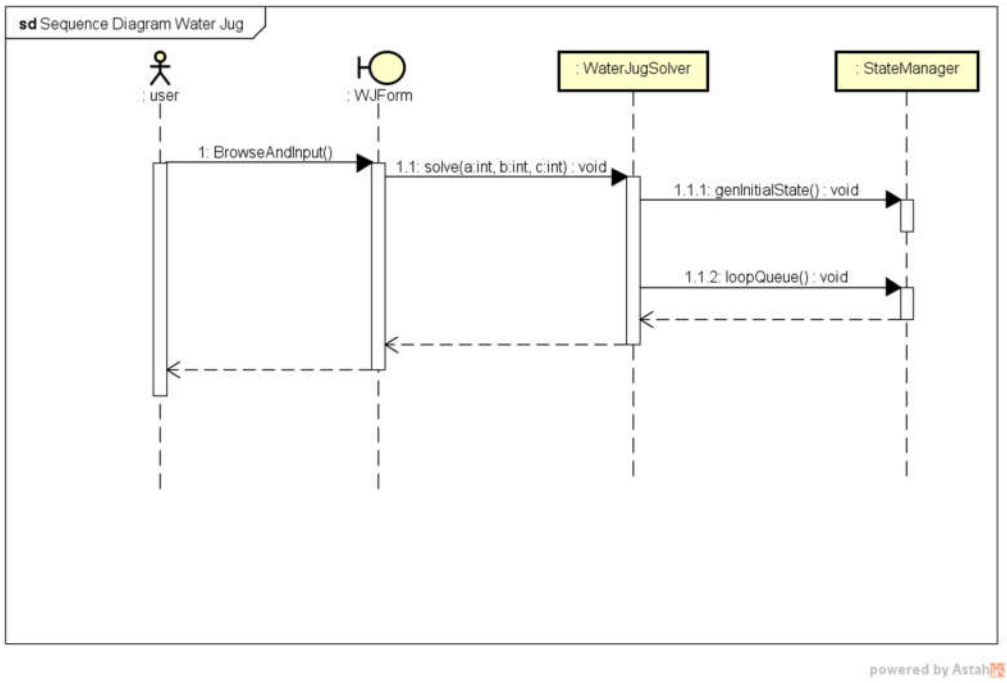
There are two jugs having capacities *a* and *b* liters. There is a tank with unlimited water. Given a target volumn *c* liters. Find the way to get exactly *c* liters water (*a, b, c* are positive integers).

b. Design

- Use the breadth first search technique to find the optimal move path in a state transition schema. Each state is modelled by a pair (x, y) in which x, và y respectively the amount of water in the jugs 1 and 2. The initial state is (0, 0) and the target state is (x, c) or (c, y) or (x, y) with x+y=c. From the state (x, y), neighboring states can be generated by following operations:
  - Empty the jug 1 → state (0, y)
  - Empty the jug 2 → state (x, 0)
  - Full fill the jug 1 → state (a, y)
  - Full fill the jug 2 → state (x, b)
  - Pour the jug 1 into the jug 2 until the jug 2 is full (if x+y>=b) → state (x+y-b, b)
  - Pour the jug 1 into the jug 2 until the jug 1 is empty (if x+y < b) → state (0, x+y)
  - Pour from the jug 2 into the jug 1 until the jug 1 is full (if x+y >= a) → state (a, x+y-a)
  - Pour from the jug 2 into the jug 1 until the jug 2 is empty (if x+y < a) → state (x+y,0)

- method check(int x, int y) return true if state (x, y) has been generated
- method final(int x, int y) return true if state (x, y) is a final state





12. Order Pickup

A staff need to traverse a warehouse for picking enough amount of packages of pepsi of an order. There are n positions 1, 2, …, n where the position i contains a_i packages of pepsi. The staff departs from the door of the warehouse (position 0) and have to pickup Q packages of pepsi. The travel distance from position i to position j is $d(i,j)$ (i, j = 0, 1,…, n). Find the way for the staff visiting positions of the warehouse to pickup enough Q packages of pepsi and come back to the door in minimal travel distance.

Input file:

- Line 1: n and Q
- Line 2: a_1, a_2, …, a_n
- Line i+3 (i = 0, 1,…, n): write the i^{th} line of the distance matrix $d(i,0)$, $d(i,1)$, …, $d(i,n)$

Output file: Write a unique number which is the minimal distance found.

Example:

Input	Ouput
5 10 **3 6 2 4 1** **0 3 6 4 6 2** **3 0 7 8 2 1** **6 7 0 1 4 9** **4 8 1 0 3 4** **6 2 4 3 0 1** **2 1 9 4 1 0**	**12**

Optimal route for picking up order is: 0 – 1 – 5 – 4 – 3 – 0 với độ dài bằng $d(0,1) + d(1,5) + d(5,4) + d(4,3) + d(3,0) = 3 + 1 + 1 + 3 + 4 = 12$ và tổng lượng hàng tại các điểm trên đường đi là $a_1 + a_5 + a_4 + a_3 = 1 + 1 + 4 + 2 = 10$

13. DataBase

a. Problem description

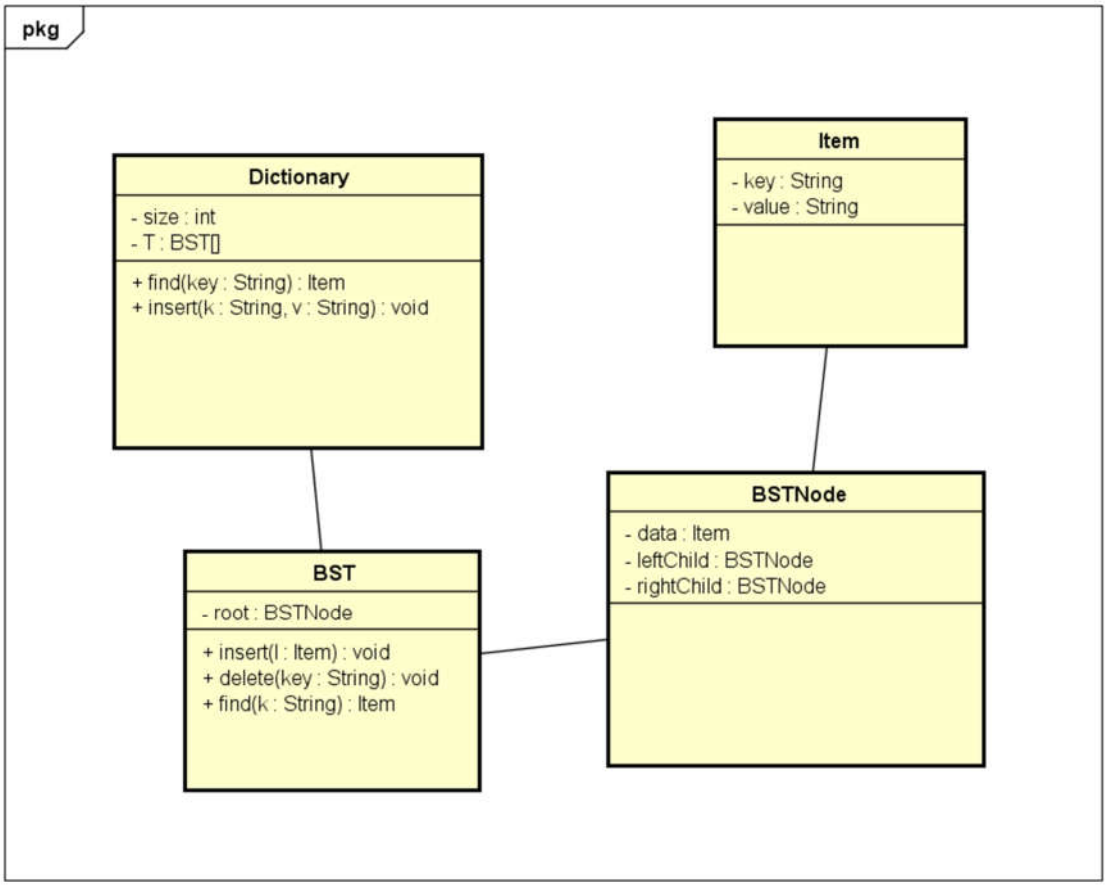
Build a data structure (database) storing profiles of staffs in an organization. Each staff has a unique ID which is a string and other personal information. The data structure allows users to perfom following operations

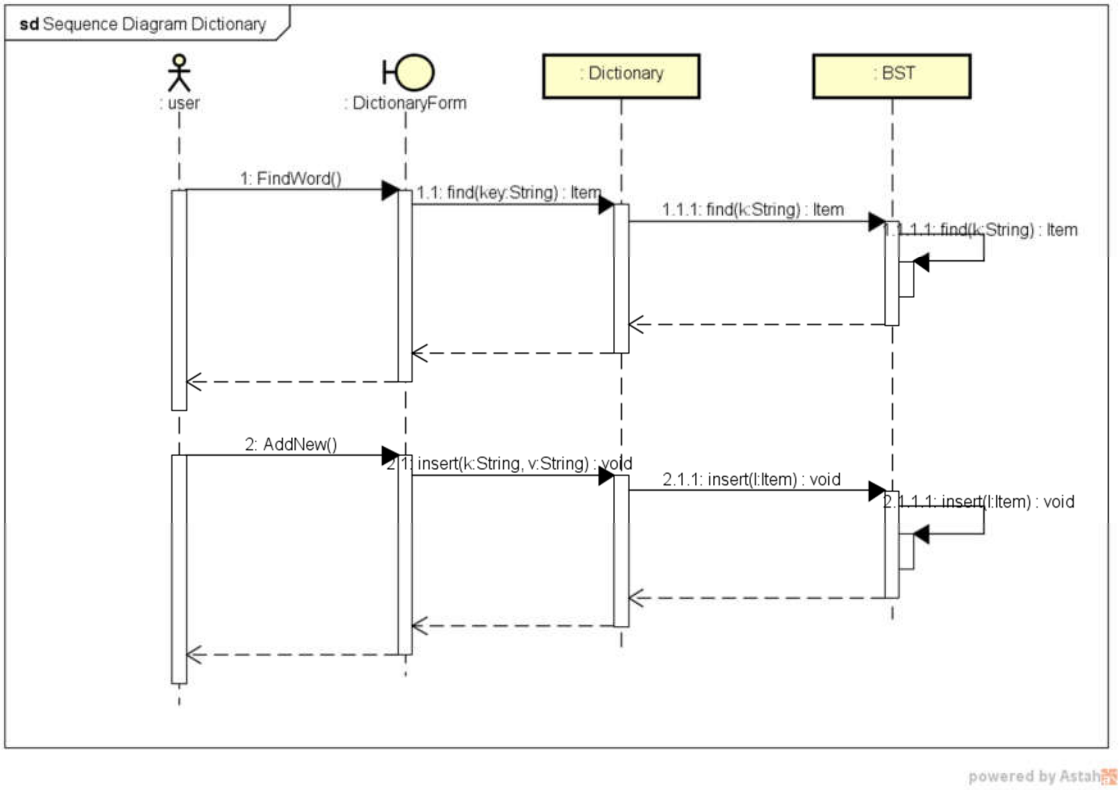
- add(String ID, Info I): add a profile of a staff to the database
- find(String ID): find and return the profile of the staff given his/her ID
- delete(String ID): remove the profile of the staff given his/her ID

The data structure employ hashing and binary search tree techniques.

b. Design

- Class Item represents the profiles of staffs
- Class BST represents binary search trees
- Lớp BSTNode represents nodes of the BST. It contains an Item and references to the left and right children.
- Class Dictionary represents the data structure using the hashing technique and binary search trees for separate chaining.





14. Store and Search keys in Database

A database contains a sequence of key k1, k2, ..., kn which are strings (1<=n<=100000). Perform a sequence of actions of two kinds:

- $find k: find and return 1 if k exists in the database, and return 0, otherwise
- $insert k: insert a key k into the database and return 1 if the insertion is successful (k does not exist in the database) and return 0 if the insertion is failed (k exists in the database)

Note that the length of any key is greater than 0 and less than or equal to 50.

Hint: implementation using Hashing and Binary Search Tree

Input
Two blocks of information. The first block contains a key of (k1,k2,...,kn) in each line. The first block is terminated with a line containing *. The second block is a sequence of actions of two finds described above: each line contains 2 string: cmd and k in

which cmd = $find or $insert and k is the key (parameter of the action). The second block is terminated with a line containing ***. Note that the number of actions can be up to 100000.

Output
Each line contains the result (0 or 1) of the corresponding action.

Example

input
```
computer
university
school
*
$find book
$find university
$insert book
$find book
$insert university
***
```

output
```
0
1
1
1
0
```

15. Tree Manipulation

Perform a sequence of actions including building the tree and counting specific nodes of the built tree. The input contains, in each line, a command of type:

- $MakeRoot u: create a root node u of the tree
- $Insert u v: create a new node u and insert it at the end of the children nodes of v
- $Height u: compute the height of the node u
- $Depth u: compute the depth of the node u

Input

The first line always is the command $MakeRoot u: create a root node u of the tree.

In subsequent lines, each line is a command of type

- $Insert u v: create a new node u and insert it at the end of the children nodes of v
- $Height u: compute the height of the node u
- $Depth u: compute the depth of the node u

Output

Each line contains the number of leaf nodes of the corresponding query $Height and $Depth described above.

Example

input

```
$MakeRoot 10
$Insert 11 10
$Insert 1 10
$Insert 3 10
$Insert 5 11
$Insert 4 11
$Height 10
$Depth 10
$Insert 8 3
$Insert 2 3
$Insert 7 3
$Insert 6 4
$Insert 9 4
$Height 10
$Depth 10
$Depth 3
*
```

output

```
3
1
4
1
2
```

16. COVID19

Information about COVID19 is collected and stored in a database, each object consists of following information:

- Day: 1, 2, 3, …
- Region code (string of a,…,z, A,…,Z and 0,…,9): codes of regions must be pairwise different
- Number of patients (integer) reported

Perform a sequence of commands (update actions and queries) described as follows

Index	Command	Description
1	$Update <region_code> <day> <number_of_patients>	Update information about new number of patients <number_of_patients> reported on day <day> in the region <region_code> In other words, this action inserts a record (<region_code>, <day>, <number_of_patients>) into database if the pair (<region_code>, <day>) does not exist *Note: if the pair (<region_code>, <day>) exists, then do not update*
2	$CountTotalPatients	Count and print to stdout (in a new line) the total number of patients reported until the current moment
3	$FindNumberPatients <region_code> <day>	Find and print to stdout (in a new line) the number of patients reported on day <day> in the region <region_code>
4	$CountNumberPatientsOfPeriod <start_day> <end_day>	Count and print to stdout (in a new line) the number of patients reported in all regions from day <start_day> until day <end_day>

5	$CountNumberPatientsOfRegion <region_code>	Count and print to stdout (in a new line) the number of patients reported in the region <region_code>

Number of distinct region codes can be upto 1000, and the number of days can be upto 200.

Input

- Each line contains a command with above format, terminated by a line ***

Output

- Each line contains a result printed out (by commands 2, 3, 4, 5 described in the table above)

Example

Stdin	Stdout
$Update HN0001 4 20 $Update HN0002 2 40 $Update HN0001 2 30 $Update HN0002 1 10 $CountTotalPatients $Update HN0003 1 45 $CountNumberPatientsOfPeriod 1 3 $CountNumberPatientsOfRegion HN0002 $FindNumberPatients HN0003 2 $FindNumberPatients HN0003 1 ***	100 125 50 0 45