**Java - Competitive Programming Challenge**

**Question 4**

There will be a list of items in the form of a 2-dimensional string array where each element contains [name, relevance, price]. Given the sort column, the sort order (0: ascending, 1: descending), the number of items to be displayed on each page (except for the last page which may have fewer), and a page number, determine the list of item names in the specified page while respecting the item's order. Page numbering starts at 0.

**Example**

items = [['item1', '10', '15'], ['item2', '3', '4'], ['item3', '17', '8']]

sortParameter = 1

sortOrder = 0

itemsPerPage = 2

pageNumber = 1

n = 3 items

Sort them by (relevance: 1) in ascending order (items = [['item2', '3', '4'], ['item1', '10', '15'], ['item3', '17', '8']])

Display up to 2 items in each page

The page 0 contains 2 item names ['item2', 'item1'] and page 1 contains only 1 item name, so result = 'item3'

**Function Description**

Complete the function fetchItemsToDisplay in the editor below.

fetchItemsToDisplay has the following parameter(s):

* string items[n][3]: a 2D array of arrays of strings in the form [name, relevance, price] int sortParameter: the column of the items to sort on
* int sortOrder:0 = ascending and 1 = descending int itemsPerPage: the number of items per page
* int pageNumber: the page number to display item names

**Returns**:

string pageItems[m]: array of item names on the requested page in the order they are displayed

**Constraints**

* 1 ≤ n < 105
* 1 ≤ m ≤ n
* 0 ≤ relevance, price < 108
* relevance and price are both integers 1 ≤ itemsPerPage < 20
* 0 ≤ pageNumber < 10

**Question 5**

**Question Description**

FC Codelona is trying to assemble a team from a roster of available players. They have a minimum number of players they want to sign, and each player needs to have a skill rating within a certain range. Given a list of players' skill levels with desired upper and lower bounds, determine how many teams can be created from the list.

**Example**

skills = [12, 4, 6, 13, 5, 10]

minPlayers = 3

minLevel = 4

maxLevel = 10

The list includes players with skill levels [12, 4, 6, 13, 5, 10].

They want to hire at least 3 players with skill levels between 4 and 10, inclusive. Four of the players with the following skill levels { 4, 6, 5,10} meet the criteria.

There are 5 ways to form a team of 3 players : {4, 5, 6}, {4, 6, 10}, {4, 5,10}, {5, 6, 10}, and {4, 5, 6, 10}.

Return 5.

**Function Description:**

Complete the function countTeams in the editor below.

countTeams has the following parameter(s):

* int skills[n]: an array of integers that represent the skill level per player
* int minPlayers: the minimum number of team members required
* int minLevel: the lower limit for skill level, inclusive
* int maxLevel: the upper limit for skill level, inclusive

**Return:**

* int: the total number of teams that can be formed per the criteria

**Constraints**

* 1 ≤ n ≤ 20
* 1 ≤ minPlayers ≤ n
* 1 ≤ minLevel ≤ maxLevel ≤ 1000
* 1 ≤ skills[i] ≤ 1000

**Question 7:**

**Question Description:**

Pat is an ordinary kid who works hard to be a great runner. As part of training, Pat must run sprints of different intervals on a straight trail. The trail has numbered markers that the coach uses as goals. Pat's coach provides a list of goals to reach in order. Each time Pat starts at, stops at, or passes a marker it is considered a visit. Determine the lowest numbered marker that is visited the most times during Pat's day of training.

Example

n = 5

sprints = [2, 4, 1, 3]

if the number of markers on the trail, n = 5, and assigned sprints = [2, 4, 1, 3], Pat first sprints from position 2 → 4. The next sprint is from position 4 → 1, and then 1 → 3. A marker numbered position p is considered to be visited each time Pat either starts or ends a sprint there and each time it is passed while sprinting. The total number of visits to each position in the example is calculated like so:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Total Visits Per Position | | | | | |
| Sprint | 1 | 2 | 3 | 4 | 5 |
| 2 → 4 |  | ☺→ | → | →☺ |  |
| 4 → 1 | ☺← | ← | ← | ←☺ |  |
| 1 → 3 | ☺→ | → | →☺ |  |  |
| Total Visits | 2 | 3 | 3 | 2 | 0 |

Pat has visited markers 2 and 3 a total of 3 times each. Since 2 < 3, the lowest numbered marker that is visited the most times during Pat's day of training is 2.

**Function Description:**

Complete the function getMostVisited in the editor below.

getMostVisited has the following parameter(s):

* int n: an integer denoting the number of markers along the trail
* int sprints[m]: an array of integers denoting the sequence of markers to reach, beginning at the marker shown in sprints[0].

**Returns:**

int: an integer denoting Pat's most visited position on the trail after performing all m − 1 sprints. If there are multiple such answers, return the smallest one.

**Constraints:**

* 1 ≤ n ≤ 105
* 2 ≤ m ≤ 105
* 1 ≤ sprints[i] ≤ m (where 0 ≤ i < m) sprints[i-1] ≠ sprints[i] (where 0 < i < m)