CS102 - Algorithms and Programming I
Lab 06

Lab Objectives: Recursion.

For all labs in CS 102, your solutions must conform to the CS101/102 style guidelines.

LAB RULES:

- You are allowed to refer to your lecture notes and textbooks during lab sessions. You are not allowed to talk or exchange any form of information with your classmates. All solutions should be your own. If you have questions, please consult your TAs or tutors.
- When you are done your TA will ask you questions about the assignment and have you sign the attendance sheet. You must attend the lab session and show your code to the assistants to receive a grade for the lab.

1. Helen is a hard worker, and she has decided to run a programming tournament this time. This competition has some weird rules. There are 2 participant teams where team i has coding ability or power. At each round of tournament, Helen divides teams into $n\ p\ i$ two parts where the first part consists of the first half remaining teams and the second part consists of the second half of the remaining team in the tournament.

   At each round, Helen decides to continue working with teams of one part and say goodbye to the teams in the other part. This is done while she gives award to the most powerful team of the part she said goodbye to. This award is worthy of that specific team's power. Helen likes to give the maximum possible awards to participants. To do so, she needs to find out the amount of budget needed for total awards. Note that she also needs to give an award to the last remaining team.

   For example, if there are 8 teams with powers 4,1,8,1,7,1,6,1, Helen can say goodbye to the first four teams (teams 1,2,3,4) in first round and give a reward worthy of 8 to team 3. Then she would continue with saying goodbye to two teams in the second part (teams numbered 7 and 8) and give a award worthy of 6 to team 7. In the next round, she would say goodbye to team 5 and give award 7 to them. Finally, she would give the last award (worthy of 1) to team 6. Thus, the total amount of award she would need would be22 = 1 + 7 + 6 + 8. In another scenario, she could have continued dividing teams by saying goodbye to the second 4 teams, then teams 3 and 4, and finally to team 1 which would lead to total award amount of 20 = 1 + 4 + 8 + 7. Write a **recursive** program that calculates the total budget needed for awards.

   Restrictions: 1 <= n <= 17 1 <= $p\ i$ <= 10 9 3.

2. Helen is working in a cheese factory where her income is s at ith month ( ). She has decided to increase her income i 1 <= i <= n at each month in a way that her income at each month should be the sum of incomes for all previous months.
   In other words:
   - Income at month n + 1 or s is equal to n+1 s 1 + s 2 + ... + s n
   - Income at month n + 2 or s is equal to n+2 s 1 + s 2 + ... + s n + s n+1
   - Income at month n + 3 or s is equal to n+3 s 1 + s 2 + ... + s n+1 + s n+2
   - …

   Write a **recursive** program that finds Helen's income at month k or . As the resulting number might be large, find its remainder $j\ s\ k\ j$ to 10 + 7. The program should take n in first line which shows the number of replacements. In the next line, 2 integers are given n where ith integer is representative of p_i

   Input: n and q are given in the first line of input which show the number of month that Helen has got income until that point (1 <= n) and the number of questions that will be asked from you (which shows the month for which income should be calculated), respectively.

   In the second line, n positive integers s are given which represent her income at each month of n months (1 , s 2 , ..., s n 1 <= s). i <= 100

   In the next q lines a positive integer k is given at each line which asks you to find Helen's income in month to modulo . j $k\ j$ 10 9 + 7 The output should be in q lines which show the answer (income of a specific month) of questions at each q lines.

   Restrictions: n + 1 <= $k\ j$ <= 10 9 4.

3. Renowned scientific research company Aperture Science is trying to develop a portal technology. Help them by extending the recursive definition of Maze traversal code that we provided (MazeSearch.java and Maze.java by Lewis/Loftus) such that the agent can jump between portals. Each portal has two sides. If the agent steps into one side of the portal it will be teleported to the other side. That way, even if there is no road between two locations, it can reach its destination by jumping between portals. Note that there can be multiple portals. Also, the agent always starts from the top-left corner and its destination is bottom-right corner. At the end of the day, you should create a recursive program that traverses the Maze and tries to find if reaching the destination is possible by jumping through portals and walking.

   Example:
   ```
   1, 0, 0, 0, 0, 0
   1, 1, 1, 0, 0, 0
   0, 0, 0, 0, 1, 1
   ```

   Assume that above matrix is a top-down view of the maze. The agent starts at top-left location (row=0, column=0) and its destination is bottom-right location (row=2, column=5). 1 signifies walkable areas and 0 means walls. So, in this case, it cannot reach its destination since there is no path of 1s between them.

   Let's add portals,
   ```
   1, 0, 0, 0, 0, 0
   1, 1, 1, 0, 0, 0
   0, 0, 0, 0, 1, 1
   ```

   Now, **blue** locations (row=1, column=2 and row=2, column=4) are connected through portal. Therefore, it can jump between them to reach the destination.

   Test Cases:

   Q1.
   input 1: 2
   1 2 3 4
   output 1: 9
   inout 2: 3
   1 6 1 7 1 8 1 4
   output 2: 22

   Q2.
   input: 3 2
   1 2 3
   4
   5
   14
   output:
   6
   12
   ?

   Q3.
   Color code:
   red, green, brown = different portals (i.e., two green locations mean they are connected through a portal)
   ```
   1, 0, 0, 0, 0, 0
   1, 1, 1, 0, 0, 0
   0, 0, 0, 0, 1, 1
   => Can reach end
   ```

```
1, 1, 0, 0, 0, 0
1, 1, 1, 1, 1, 0
1, 1, 1, 1, 1, 0
0, 0, 0, 1, 1, 1
=> Can reach end

1, 1, 1, 1, 1, 1
0, 1, 1, 1, 1, 1
0, 1, 0, 0, 0, 0
0, 0, 0, 1, 0, 1
=> Cannot reach end

1, 1, 0, 1, 1, 1
0, 0, 0, 0, 0, 1
0, 0, 0, 1, 0, 0
0, 0, 0, 0, 0, 0
1, 1, 0, 0, 1, 1
=> Can reach end
```