Lab Assignment 9
Arrays and recursion
COL100

May 14, 2021

1 Sherlock and Array

Watson gives Sherlock an integer array A of size n. His challenge is to find if there is an element in the array such that the sum of all elements to its left is equal to the sum of all elements to its right. Write a function *findElement(int A[], n)* that takes Array A and integer n as input and return 1 if such element exists else return 0.

1.1 Example:

Input :

```
3  1  2  3
```

Output :

```
0
```

Explanation : No such element is there.

1.2 Example:

Input:

```
4  1  2  3  3
```

Output :

```
1
```

Explanation : sum of all elements to the left of the 3^{rd} element of array is 3 and sum of all elements to right of the 3^{rd} element of array is also 3 so function returns 1.

2 Peak Index in a Mountain Array

Let's call an array A of size n, a *mountain* if the following properties hold:

- $n >= 3$

- There exists some i with $0 < i < n - 1$ such that:
  - $A[0] < A[1] < ... < A[i - 1] < A[i]$
  - $A[i] > A[i + 1] > ... > A[n - 1]$

You are given an array A of size n that is guaranteed to be a mountain. Write a function *peakIndex(int A[], int n)* that takes as parameters array A and integer n and returns any i such that $A[0] < A[1] < ... < A[i - 1] < A[i] > A[i + 1] > ... > A[n - 1]$. Use ideas involved in Binary Search to solve this problem.

2.1 Example 1:

Input:

```
0  1  0
```

Output:

```
1
```

Explanation: $A[0] < A[1] > A[2]$

2.2 Example 2:

Input:

```
4
0  10  5  2
```

Output:

```
1
```

Explanation: $A[0] < A[1] > A[2] > A[3]$

3 Ice Cream Parlor

Two friends like to pool their money and go to the ice cream parlor. They always choose two distinct flavors and they try to spend all of their money. Given an array A containing prices for different flavors of ice cream, select the *two* that will cost all the money they have. Print the indices of the flavors of ice-cream that they will choose. Following constraints are applicable -

- It is guaranteed that they will be able to spend all their money
- Nested loops are not allowed

Input Format: Take integer m as input which denotes total money both friends will have. Take integer n as input which denotes total flavors of ice-cream present. Take elements of Array A as an input denoting prices for different flavors of ice cream.
Output Format : Print the *indices* of the flavors of ice-cream that they will choose. In case of multiple pair, print anyone of them.

3.1 Example 1:

Input:

```
4
5
1  4  5  3  2
```

Output:

```
0  3
```

Explanation: $A[0] + A[3] = 4$

3.2 Example 2:

Input:

```
4
4
2  2  4  3
```

Output:

```
0  1
```

Explanation: $A[0] + A[1] = 4$

4 Game of thrones

Dothraki are planning an attack to usurp King Robert's throne. King Robert learns of this conspiracy from Raven and plans to lock the single door through which the enemy can enter his kingdom. But to lock the door he needs a key that is an *anagram* of a *palindrome*. He starts to go through his box of strings, checking to see if they can be rearranged into a palindrome.

You are given a string, write a function *possiblePal(char str[])* which takes string as an input and returns *YES* if the string can be rearranged into a palindrome else *NO*. String will contain lowercase character only.

4.1 Example 1:

Input:

```
aaabbbb
```

Output:

```
YES
```

Explanation: The rearranged palindrome for the given string is *bbaaabb*

4.2 Example 2:

Input:

```
cdcdcdcdeeeef
```

Output:

```
YES
```

Explanation: The rearranged palindrome for the given string is *ccddeefeeddcc*

5 Best Time to Buy and Sell Stock

You are given an array prices where $prices[i]$ is the price of a given stock on i^{th} day. You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock.

Write a function *maxProfit(int prices[], int n)* that takes array prices and integer n as input and returns the maximum profit you can achieve from the transaction mentioned above. If you cannot achieve any profit return 0.

5.1 Example 1:

Input:

```
6
7 1 5 3 6 4
```

Output:

```
5
```

Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5. Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.

5.2 Example 2:

Input:

```
5
7 6 4 3 1
```

Output:

```
0
```

Explanation: In this case, no transactions are done and the max profit = 0.

6 String to integer (atoi)

The standard function used to convert a string of numbers to integers is *atoi()* which is part of *stdlib* library. Your task is to write a custom function *int myatoi(char A[], int length)* which will return integer from the given string of numbers. Implementation need to be done using recursion, no loops allowed. In *main()* function, you need to scan input as string. String should contain only integer values, no special characters or alphabet.

6.1 Example 1:

Input:

```
4
1123
```

Output:

```
1123
```

6.2 Example 2:

Input:

```
4
112a
```

Output:

```
Invalid string
```

7 Candy Game

Game start when user enters some number of candies n. You can then give back some candies, but you must follow these rules -

- If n is even, then you may give back exactly $\frac{n}{2}$ candies.
- If n is divisible by 3 or 4, then you may multiply the last two digits of n and give back this many candies.
- If n is divisible by 5, then you may give back exactly 42 candies.

The aim of the game is to end up with exactly 42 candies. Your task is to write a recursive function *bool candygame(int n)* which will return *true* if it is possible to win game with n candies otherwise *false*. Implementation need to be done using recursion, no loops allowed.

7.1 Example 1:

Input:

```
250
```

Output:

```
True
```

Explanation:

- Initial candies 250
- Divisible by 5, return 42 candies. Leftover is 208.
- As 208 is even, return 104 candies. Leftover is 104.

- As 104 is even, return 104 candies. Leftover is 52.
- As 52 is divisible by 4, return 10 candies. Leftover is 42. Here you can also follow the divisible by 2 line but that will make candies less then 42 so you lost the game.
- Hurray, you have won the game

7.2 Example 2:

Input:

```
85
```

Output:

```
False
```

Explanation:

- Initial candies 85
- Divisible by 5, return 42 candies. Leftover is 43.
- 43 is prime, you lost the game.

8 Inverse of Matrix

Given NxN square matrix, write a program to find the inverse if it exists.
Input: N, Followed by N lines with integers
Output: NxN inverse matrix Or Inverse does not exist
To calculate the inverse of matrix A use the following -

$$A^{-1} = \frac{1}{det(A)} * adj(A)$$

Where, $det(A)$ is determinant of A and $adj(A)$ is adjoint matrix of the A. Inverse will exist only if det(A) is not equal to 0.
For more details on adjoint and inverse of the matrix: https://nptel.ac.in/content/storage2/courses/122104018/node29.html

8.1 Example 1:

Input:

```
3
3 0 2
2 0 -2
0 1 1
```

Output:

```
0.20 0.20 0.00
-0.20 0.30 1.00
0.20 -0.30 0.00
```

Explanation: Cofactors of given matrix:

- Cofactor C_{00} = 0-(-2) = 2
- Cofactor C_{01} = -1*(2-(0)) = -2
- Cofactor C_{02} = 2-(0) = 2
- Cofactor C_{10} = -1*(0-(-2)) = -2
- Cofactor C_{11} = 3-(0) = 3
- Cofactor C_{12} = -1*(3-(0)) = -3
- Cofactor C_{20} = 0-(0) = 0
- Cofactor C_{21} = -1*(-6-(4)) = 10
- Cofactor C_{22} = 0-(0) = 0

Adjoint matrix B will be

$$\begin{matrix} 2 & 2 & 0 \\ -2 & 3 & 10 \\ -2 & -3 & 0 \end{matrix}$$

And $det(A) = 10$

9 Rotated Matrix

Given NxN matrix, output the matrix rotated by 90^0 degree (clockwise)
Constraint: Not allowed to use extra 2D matrix. This type of operation is called **inplace** rotation.
Hint: Rotated matrix = transposed matrix + reflected/mirrored matrix

9.1 Example 1:

Input:

```
3
1  2  3
4  5  6
7  8  9
```

Output:

```
7  4  1
8  5  2
9  6  3
```

10 Zigzag (or diagonal) traversal of Matrix

Given a MxN matrix, print all elements of the given matrix in diagonal order.

10.1 Example 1:

Input:

```
5 4
 1  2  3  4
 5  6  7  8
 9 10 11 12
13 14 15 16
17 18 19 20
```

Output:

```
1
5 2
9 6 3
13 10 7 4
17 14 11 8
18 15 12
19 16
20
```

Challenge Problems

11 Minimum Sum Path in a Triangle

Given a triangular structure of numbers, find the minimum path sum from top to bottom.
Constraints:

- There exist only one minimum path
- The path must contain exactly one element from each row of triangle
- The input first line will be height of the triangle followed by elements in each row

11.1 Example 1:

Input:

```
4
3
6 4
5 2 7
9 1 8 6
```

Output:

```
10
```

Explanation : Minimum path sum is **3 + 4 + 2 + 1 = 10**

```
   3
  6  4
 5  2  7
9  1  8  6
```

12 Fractal patterns

Write a recursive function *void pattern(unsigned int n, unsigned int i)* to print a fractal pattern, the longest line consist of the n stars and beginning in the i^{th} column. Length n should be power of 2. Can you find two smaller versions of the pattern within the large pattern?

12.1 Example 1:

Input:

```
8  0
```

Output: pattern(8,0)

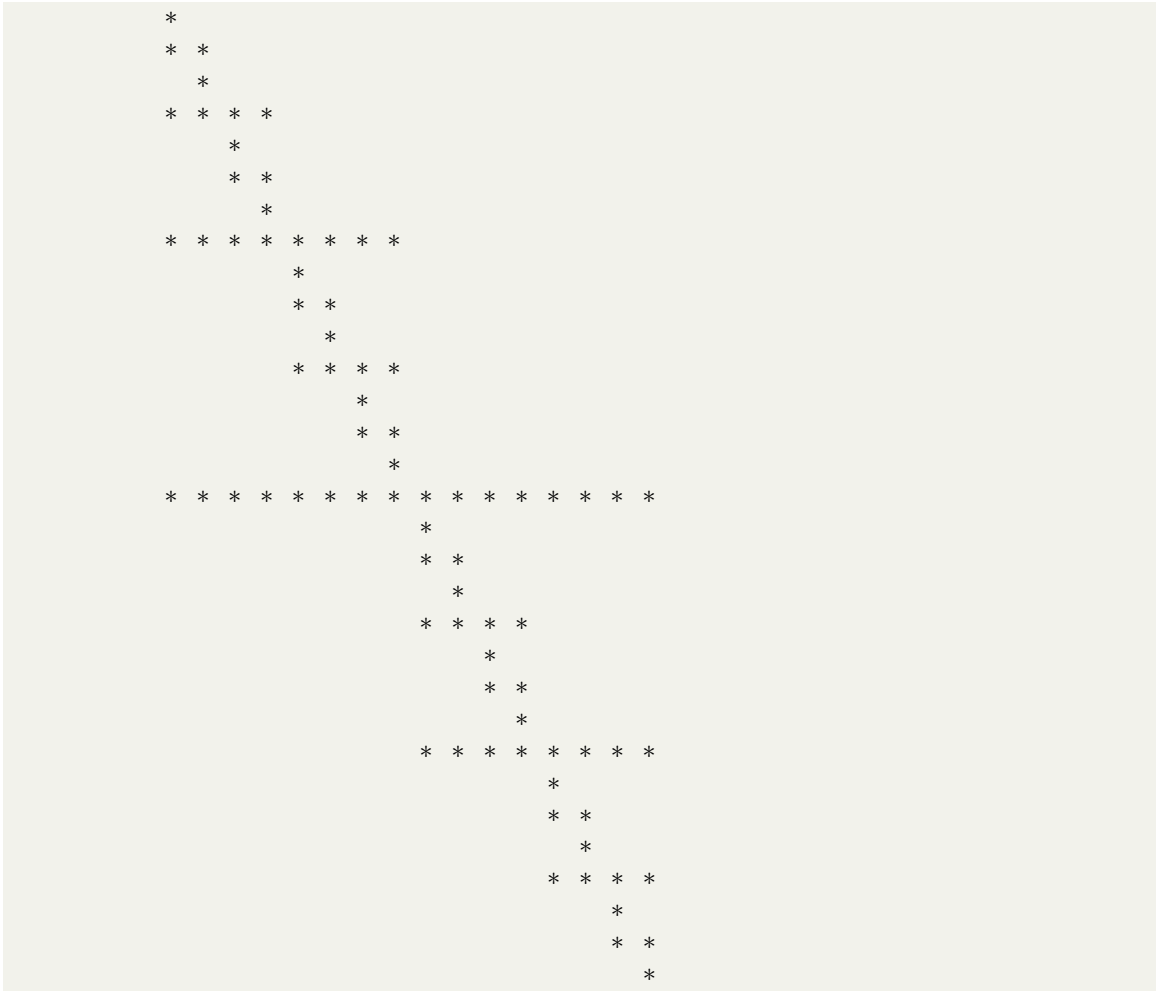
```
*
* *
  *
* * * *
    *
    * *
      *
* * * * * * * *
        *
        * *
          *
        * * * *
            *
            * *
              *
```

12.2 Example 2:

Input:

```
16  5
```

Output: pattern(16,5)



Submission and other logistics

Submit at least 4 solutions (from the first 10 questions) as a zip file on Gradescope (to your respective group's course). There is no need to submit solutions to the challenge problems. Submit only one .c file for each question. Use separate .c files for each new question. Please name your .c files as per the question number (q1.c, q2.c, ... etc). Following this naming convention will help TAs to figure out where to look for the answers easily. Also add the corresponding screenshots, showing output of your code on certain input examples. You can also submit more than 4 or all questions to increase your chances of full-marks.

Example: To zip folder 'a9' as 'a9.zip':

```
zip -r a9.zip a9
```

It is highly **recommended** that you name the code files and variables in those code files with proper names as per the question to easily identify them. Comments in your codes are also highly encouraged and makes life easier for everyone.
You can check **2nd Chapter** in NASA's C style guide for styling recommendations.

You can work either individually or with another student of your group for the assignment.only one submission on gradescope is enough for a team but you need to add your teammate on gradescope after submission.

Follow these steps for adding your team member
Note: you can change your team for future assignments