7Vals Test 2021

Question 9

Question text

Assume **x** is a variable that represents a data structure which is initially empty. The following operations are performed on **x**:

- x.push(5)
- x.push(13)
- x.push(19)
- x.pop() // 5 is popped
- x.push(15)
- x.pop() // 13 is popped

Given this information, which of the following data structures could **x** belong to? Select one or more:

- ☐ a. Heap
- ☐ b. Array
- ☐ c. Stack
- ☑ d. Queue

Question 10

Question text

Two sorted arrays of size **n** can be combined to create a single sorted array **optimally** in:

Select one:

- ○ a. O(lg(n))
- ◉ b. O(n)
- ○ c. O(n*lg(n))
- ○ d. O(n^2)

Question 11

Question text

Analyze the following piece of code and determine its time and space complexity.

```
int x = 0, y = 0;

for (i = 0; i < V; i++) {
   x = x + x;
}

for (j = 0; j < W; j++) {
   y = y + y;
}
```

Select one:

- a. O(V + W) time, O(V + W) space
- b. O(V * W) time, O(N + M) space
- c. O(V + W) time, O(1) space
- d. O(V * W) time, O(1) space

Question 12

Question text

How many times will '@' be printed when bar(4) is called? (Assume there are no syntax errors in the given code)

```
void bar (int i)
{
   if (i > 1)
   {
      bar(i / 2)
      bar(i / 2)
   }
   print('@')
}
```

Select one:

- a. None of the above
- b. 8

- c. 3
- d. 7
- e. 4

Question 13

Question text

What is the time complexity of the following piece of code?

```
for (int j = 0; j < 10; j++)
   for (int k = 0; k < N; k++)
      for (int m = N - 2; m < N + 2; m++)
         cout << "@" << endl;
```

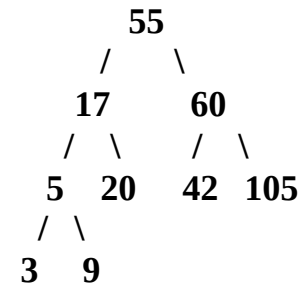
Select one:

- a. O(N * lg(N))
- b. O(N)
- c. O(N^2)
- d. O(N^3)

Question 14

Question text

Is this a valid binary search tree?



Select one:

- True
- False

Question 15

Question text

Write a function **string* retrunEfficientPersons(string names[], int workHours[], int faultRates[], int dataLength)** that **returns an array of strings representing the most and least efficient persons.** The input arguments are described below.

names - An array of strings as person names.

workHours - An array of integers, the time taken by each person to complete the work.

faultRates - An array of integers, the amount of work not done by each person (in percentage)

dataLength - An integer, representing the length of given arrays (all arrays have same length)

Explanation:

Suppose there are 5 persons named Ahmed, Hamza, Aleem, Mujeeb, Mujtaba. All are assigned the same amount of work. They complete work in 2, 3, 4, 5, 6 hours and the fault rate (amount of work not done) among them is 30%, 40%, 50%, 10%, 15%. Write a function that returns the most efficient person and the least efficient person. Note that for simplicity assume that every piece of work has same difficulty level and takes constant time.

Note:

1) If there are persons with same efficiencies your result should contain the person coming first in the array.

2) If its not possible to compute the result return NULL.

Example:

INPUT:

names = { "ahmed", "hamza", "aleem", "mujeeb", "mujtaba" }

workHours = { 2, 3, 4, 5, 6 }

faultRates = { 30, 40, 50, 10, 15 }

OUTPUT:

['ahmed', 'aleem']

Question 16

Question text

Due to Covid-19, Nando's have fixed working hours with proper social distancing all around the restaurant premises. Considering this we have 5 different parameters for restaurant as TotalCustomers,
WorkingHoursPerDay, PersonsAllowedToSitOnASingleTable,
ReservationTimePerTable, TotalTables.

Description For Parameters:
1) TotalCustomers (count provided for pre-booking), integer i.e. 100, 200, 500 etc.
2) WorkingHoursPerDay (Total hours allowed by Government), integer
i.e. 8,10 hours etc.
3) PersonsAllowedToSitOnASingleTable, integer i.e. 2, 3 etc.
4) ReservationTimePerTable, integer i.e. 1, 2, 3 hours etc.
5) TotalTables, integer i.e. 5,15,25 etc.

You have to write a function that can help NANDO'S to answer the following questions to arrange the event of the details provided before to them:

1. IS IT POSSIBLE TO SERVE ALL PERSONS (TOTAL CUSTOMERS) IN A DAY?
2. HOW MANY PERSONS WILL BE SEATED OUT OF TOTAL CUSTOMERS IN A DAY?

3. HOW MANY PERSONS WILL BE REFUSED OUT OF TOTAL CUSTOMERS IN A DAY?
4. HOW MUCH TIME IS REQUIRED TO FULFILL ALL CUSTOMERS? TOTAL TIME CAN BE MORE THAN 1 DAY'S WORKING HOURS.

write a function **int* restaurantAnswers(int totalTables, int personsAllowedToSitOnTable, int workingHours, int reservationTimePerTable, int totalClients)** that **returns an integer array** with first value as a result of 1st question, 2nd value as a result of 2nd question, 3rd value as a result of 3rd question and 4th value as a result of 4th question. The result of first question is a boolean value so you can use 0/1 to represent the boolean value.

Example 1:

INPUT:

totaltables = 5
personsAllowedToSitOnATable = 5
workingHours= 2
reservationTimePerTable = 1
totalClients = 100

OUTPUT:

[0, 50, 50, 4]

Example 2:

INPUT:

totaltables = 5
personsAllowedToSitOnATable = 10
workingHours= 10
reservationTimePerTable = 1
totalClients = 100

OUTPUT:

[1, 100, 0, 2]

Question 17

Question text

You are a character in a video game that needs to traverse a **grid** (2D array) of size **n x n**, starting from the top left of the grid and ending at the bottom right, i.e. you start at **grid[0][0]** and must end your journey at **grid[n-1][n-1]**. On each cell, you can only move either to the cell directly to the right or to the cell directly at the bottom. Concretely, if you are on **grid[i][j]**, your next move can be to either **grid[i][j+1]** or to **grid[i+1][j]**.

Each cell in the grid has a certain non-negative integer number of coins. When you visit a cell in the grid, you will collect all the coins in that cell and add them to your wallet. Your goal is to find a path that will maximize the sum of coins you can collect, starting from **grid[0][0]** and ending at **grid[n-1][n-1]**, and return that value.

Input:

n - The size of the grid

grid - A 2D integer array of size **n** x **n**, where **grid[i][j]** contains the number of coins at cell **i, j**.

Output:

The maximum number of coins you can collect on any valid path from **grid[0][0]** to **grid[n-1][n-1]**.

Note:

Your solution may pass some smaller cases but may fail on some larger cases, so try to write an efficient solution.

Example:

n = 2

grid =

[[0, 2],

[3, 1]]

output: 4

Explanation: You start at **grid[0][0]** and must end at **grid[1][1]**.

- The first path you can take is as follows. You start at **grid[0][0]** which contains no coins. You then move to **grid[0][1]**, which contains 2 coins. You collect these coins and add them to your wallet. You then move to **grid[1][1]**, which contains 1 coin. The total coins you collected on this path are **3**.
- Alternatively, from **grid[0][0]**, you can move to **grid[1][0]**, which contains 3 coins, and then from **grid[1][0]** you can move to **grid[1][1]**, which contains 1 coin. The total coins you collected on this path are **4**.

The maximum number of coins you can collect on any valid path are **4** (which is the second path described above).