15-110: Principles of Computing

Homework 06

Due: 31st October, 2020 at 23:59

- You must solve the tasks **individually**.
- There are 50 points.

1. (15 points) **Daisies Field**

   Nour is the manager of a company that grows daisies for distributing to flower shops around the country. The daisies are grown in vases placed on a rectangular field (forming sort of a grid), with R rows and C columns of vases. Since this is a very big field, Nour has split it into $N \times M$ tiles. She made sure that N divides R and M divides C so that all the field is covered by tiles of the same size, and there is no overlap.

   Harvesting is done one tile at a time, collecting all daisies of that tile that have blossomed. Each day, Nour must decide which tile is going to be harvested, and she would like to choose the tile with the maximum number of blossomed daisies. Implement the function `daisiesField(F, n, m)` to help Nour with this task.

   The function takes as input the field of daises `F` as a list of lists. Each position indicates the number of blossomed daisies on that vase. For example, a 3x2 field where all vases have 2 blossomed daisies is represented as: `[[2,2],[2,2],[2,2]]`. The function also takes as input the number of rows n and columns m of each tile.

   Return the maximum number of daisies that can be collected that day.

   For example:

   - `daisiesField([[2,1,3],[1,1,1]], 1, 1) == 3`
   - `daisiesField([[2,1,3,4],[9,4,2,2]], 2, 2) == 16`
   - `daisiesField([[5,1,7,6],[7,4,2,6],[3,4,2,1],[8,3,5,2]], 2, 2) == 21`

2. (10 points) **Cafeteria Queue**

   Lunchtime is the most awaited time of the day for students in your school. Not only because there are no classes, but because the lunch is really good! At 12, all students run to the cafeteria and queue up for the meal.

   Professor Yilma noticed that and decided to use it to encourage students to do better in his course. He convinced the cafeteria people to serve the students not in the order they arrived, but in order of their grades, highest to lowest. This way, students with higher grades get their lunch first.

   Your task is simple: given the arrival time of the students in the cafeteria, and their respective grades in the math class, reorder the queue according to the math grades, and check how many students do not need to change place in this reordering.

   Implement the function `cafeteriaQueue(Q)` that takes as input a list of the students in the queue, identified by their grades. For example, if the first students that arrived has grade 100, the second has grade 80, and the third student's grade is 90, `Q` would be `[100,80,90]`. There will be no students with the same grade due to professor Yilma's fine grained scoring system. This function returns the number of students that *do not* need to change places in line.

   For example:

- `cafeteriaQueue([100,80,90]) == 1`
- `cafeteriaQueue([100,120,30,50]) == 0`
- `cafeteriaQueue([100,90,30,25]) == 4`

3. (10 points) **Digits count**

   Given two numbers a and b, count how many times each of the digits 0 to 9 occur in all numbers between a and b, inclusive.

   In order to make your code simpler, implement first the function `intToList(n)` that takes as input one integer `n`, and returns a list with the digits of `n` in the order they appear. For example, `intToList(1964)` should return `[1,9,6,4]`.

   Using the function `intToList`, implement the function `digitsCount(a, b)` that returns a list with 10 numbers, where position i holds the number of times digit i occurred in all numbers between `a` and `b`, inclusive. For example, `digitsCount(1, 9)` should return the list `[0,1,1,1,1,1,1,1,1,1]`.

4. (15 points) **Power Crisis**

   During a power crisis in New Zealand last winter (caused by a shortage of rain and hence low levels in the hydro dams), a contingency scheme was developed to turn off the power to areas of the country in a systematic, totally fair, manner. The country was divided up into n regions (Auckland was region number 1, and Wellington number 13). A step, s, would be picked "at random", and the power would first be turned off in region 1 (clearly the fairest starting point) and then in every s^{th} region after that, wrapping around to 1 after n, and ignoring regions already turned off. For example, if $n = 17$ and $s = 5$, power would be turned off on the regions in the order: 1,6,11,16,5,12,2,9,17,10,4,15,14,3,8,13,7.

   The problem is that it is clearly fairest to turn off the region of Wellington last (after all, that is where the Electricity headquarters are), so for a given n, the "random" step s needs to be carefully chosen so that region 13 is the last region selected.

   Your job is to write a program that determines the smallest step s that will ensure that Wellington (region 13) can function while the rest of the country is blacked out.

   For starters, implement the function `lastRegion(L, step)` that takes a list of regions `L` and a `step`, and returns what will be the last region to be turned off if that step is used. For example, `lastRegion([1,2,3,4,5], 2)` should return 2.

   Using the function `lastRegion`, implement the function `powerCrisis(n)` that takes a number `n` of regions (numbered 1 to n), and returns what is the smallest step that will result in region 13 being turned off last. The number `n` will be, of course, 13 or greater.