Unit 3 Algorithmics

Submit Task – Week 3

Choosing ADTs

Describe which ADT you would use for each problem, providing a short justification.

1. Employees of a bank are allowed to access certain functions depending on their roles. Functions include CCTV control, monitoring ATM transactions and changing customer details. How could you store the details of which employees are allowed to access which functions?
   I would use an array of dictionaries with a roles and function privilege dict and employee and role dict, This allows you to lookup any employee's role and allowed functions and name quickly.
2. At a conference, people queue to use elevators. We want to get people to their rooms as quickly and as fairly as possible. If each person is allocated a ticket when they arrive at the lobby, how could you use ADTs to work out who gets on the next elevator?
   A list of priority queues would account for multiple elevators and people with higher priority passes (eg, executives) to get to their rooms as fairly and quickly as possible
3. Reverse Polish Notation works by stating arguments and then the operation. For example:

3 5 add	3+5
2 6 add 7 times	$(2 + 6) \times 7$
2 4 add 5 3 sub exp	$(2 + 4)^{(5-3)}$

   Which ADT is most suitable for interpreting these expressions?

   A priority queue accounts for order of operations, working brackets first then indices, division/multiplication and last addition/subtraction

Prison Door Problem

A prison contains a large number of cells, numbered sequentially from 1 to 500. One night, when the prisoners are asleep, a bored guard unlocks every cell. Then, he returns to the start. He stops at every cell that is a multiple of two. If the cell is unlocked, he locks it. If it is locked, he unlocks it. He repeats this process for multiples of three, then four, and so on.

This table shows what happens for the first 7 cells.

	1	2	3	4	5	6	7
Pass 1	Unlocked	Unlocked	Unlocked	Unlocked	Unlocked	Unlocked	Unlocked
Pass 2	Unlocked	Locked	Unlocked	Locked	Unlocked	Locked	Unlocked
Pass 3	Unlocked	Locked	Locked	Locked	Unlocked	Unlocked	Unlocked
Pass 4	Unlocked	Locked	Locked	Unlocked	Unlocked	Unlocked	Unlocked

Your aim is to find out which cells are unlocked after the final pass.

1. Describe which ADTs you will use to model this problem.
   I would use a List of Dictionaries to model this problem, with each list element being a pass and each dict element being a cell with its key as the cell number and value as a Boolean lock status.
2. Write some pseudocode to perform the necessary steps.

```
Function cells(passes: INPUT, n:INPUT) { // The user should input the number
of passes the gaurd should make and the number of cells
    LIST cells := [{Cell 0: Unlocked, Cell 1: Unlocked, Cell 2: Unlocked ...
Cell n: Unlocked}]
    FOR pass_number := 0 TO passes DO {
        cells.append(DICT())
        FOR Cell := 0 TO cells.length DO {
            IF remander(Cell / (pass_number+2)) == 0 THEN {
                IF cells[pass][Cell] == Unlocked THEN {
                    cells[pass+1][Cell] = Locked
                } ELSE {
                    cells[pass+1][Cell] = Unlocked
                }
            }
            ELSE {
                cells[pass+1][Cell] = cells[pass][Cell]
            }
        }
        EndLoop
    }
    EndLoop
}
EndFunction
```

3. Translate your pseudocode into Python.
   Python file attached

4. Run your code to identify which doors are left open.
   CSV attached

Hint: You should find that 225 and 484 are open, but 170 and 499 are closed.

There is a mathematical reason behind this problem, but even if you figure it out, make sure you model the process as an algorithmics problem.