CSE102: Data Structures and Algorithm
Assignment-4
Max Marks: 60

Instructions

- Use C++ programming language only.
- Plagiarism might be checked.
- Upload a Zip folder named <RollNo>_<YourName>. If not named appropriately, it will not be evaluated.
- For the coding questions, you must ensure that your code compiles and runs otherwise these will be awarded **zero marks**.
- Students might be required to give a demo of their codes.
- `<RollNo>_Q1.cpp` - Code for Q1 (Exclude object files)
- `<RollNo>_Q2.cpp` - Code for Q2 (Exclude object files)
- `<RollNo>_Q3.cpp` - Code for Q3 (Exclude object files)
- `<RollNo>_Q4.cpp` - Code for Q4 (Exclude object files)
- `<RollNo>_Q5.cpp` - Code for Q5 (Exclude object files)
- `<RollNo>_<YourName>.pdf` - Report including answers for Q1, Q2 , Q3, Q4, Q5
- For coding questions, please provide a brief algorithm description along with an analysis of the time and space complexity.

1. (10 points) In a flooding scenario, a grid of size $m \times n$ is used to represent a region affected by floods, with 'F' denoting flooded areas and 'N' for non-flooded areas that are safe. The situation becomes critical when a region marked 'N' is surrounded by 'F' on all four cardinal directions (up, down, left, right), indicating that floodwaters fully encircle the non-flooded area. These areas cannot be saved. The task is to update the grid to show the final configuration of the flooded areas and those that remain dry. A region of N comprises all the Ns that are adjacent to another N in any of the four cardinal directions. Constraints:

   - $m ==$ `board.length`
   - $n ==$ `board[i].length`
   - $1 \leq m, n \leq 200$
   - `board[i][j]` is 'F' or 'N'.

Input:

F	F	F	F
F	N	N	F
F	F	N	F
F	N	F	F

Output:

F	F	F	F
F	F	F	F
F	F	F	F
F	N	F	F

2. (10 points) Consider an array composed of unique numbers. The objective is to identify all ordered pairs (x, y) within this array where the **remainder** of the division of the first number (x) by the second number (y) equals a specified integer, r.

   Example 1:- arr[] $= \{4, 5, 2, 1, 3\}$ and $r = 2$, the output should be $(2, 3), (2, 5), (5, 3), (2, 4)$.
   Example 2:- `arr[]` $= \{6,\ 1,\ 4,\ 10,\ 9\}$ and $r = 2$. The output is the set of pairs $(6, 4), (10, 4)$
   The length of the array n is such that $1 \leq n \leq 10^6$.
   and range of r $1 \leq k \leq 10^6$.

3. (10 points) On a 2D plane, there's a map showcasing multiple cities pinpointed by their coordinates, (x_i, y_i), listed in an array called `points`. The aim is to establish flight routes connecting these cities. The fuel needed to establish a flight route between any two cities is contingent upon their positions on the map. The cost to travel between cities is determined by the absolute differences in their x and y coordinates, expressed as $|x_i - x_j| + |y_i - y_j|$. Return the minimum fuel required to connect all locations on the map. All locations are connected if there is exactly one simple path between any two locations.
   Constraints:

   - $1 \leq$ points.length ≤ 1000
   - $-10^6 \leq x_i, y_i \leq 10^6$
   - All pairs (x_i, y_i) are distinct.

   Input :
   locations $= [[0, 0], [2, 2], [3, 10], [5, 2], [7, 0]]$
   Output :
   20
   Explanation:
   We can connect the points as shown above to get the minimum cost of 20.

   Input :
   locations $= [[3, 12], [-2, 5], [-4, 1]]$

Output :
18

4. (10 points) You need to travel from city 1 to city N across a land divided into N cities connected by M undirected roads. Each road i $(1 \leq i \leq M)$ between cities U_i and V_i has certain constraints, characterized by A_i entities each with B_i resistance points. The travel time to cross this road at time t depends on these constraints: it takes $\left\lfloor \frac{A_i}{t+1} \right\rfloor + B_i$ time units to overcome the entities and move on to city V_i, where the division is done using integer division. You can start your journey at time 0 or any integer time unit thereafter, and you can wait at any city for any integer amount of time. The goal is to minimize the overall journey time. If it is not possible to reach city N, the result should be -1. The task is to find the minimum time required to complete your journey.

   Input :

   - First line of input consists of N and M $(2 \leq N \leq 10^5,\ 0 \leq M \leq 10^5)$.
   - The next M lines contain the description of the undirected roads.
   - The i^{th} $(1 \leq i \leq M)$ line consists of U_i, V_i, B_i, A_i $(1 \leq U_i, V_i \leq N,\ 0 \leq A_i, B_i \leq 10^9)$.

   Output :

   - Print the minimum time possible according to the problem statement.

   Examples
   Input :

```
2 3
1 2 2 3
1 2 2 1
1 1 1 1
```

   Output :

```
3
```

   By selecting the second road that links city 1 to city 2 and initiating the journey from city 1 at time $t = 0$, you can arrive in city 2 with a calculated time of

$$\left\lfloor \frac{1}{0+1} \right\rfloor + 2 = 3$$

   units, where the division is integer division. Thus, the minimal travel time required to reach city 2 is 3 units.

   Input :

```
6 9
1 1  0   0
1 3  1   2
1 5  2   3
5 2 16   5
2 6  1  10
3 4  3   4
3 5  3  10
5 6  1 100
4 2  0 110
```

Output :

```
20
```

Hint :

- $t_y = t_x + \left\lfloor \frac{A_i}{t_x+1} \right\rfloor + B_i$
- t_x is the time when you depart from city ’x’ for city ’y’

Bonus :

5. (20 points) You are about to join a Guild. There are N potions in this guild, each with specific brewing requirements. You have been given a `REQUIREMENTS` list of size M, where `REQUIREMENTS`[i] consists of [’P$_i$’, ’Q$_i$’] indicating that potion P_i must be brewed before potion Q_i can be brewed.

   In one brewing cycle, you can prepare at most K potions. Thus, you must determine the minimum number of brewing cycles required to brew all potions successfully.

   Note: It is guaranteed that there is a feasible sequence to brew all the potions. The most optimal solution is not required.

   Format :

   - The first line of each test case contains three single space-separated integers ’N’, ’M’ and ’K’, representing the number of potions, number of requirements, and maximum potions that you can brew in one brewing cycle, respectively.
   - The next ’M’ line of each test case contains two single space-separated integers ’P’ and ’Q’ representing that the potion ‘P‘ must be brewed before the potion ’Q’.

   Constraints:

   - $1 \leq N \leq 15$

- $0 \leq M \leq N \cdot (N-1)/2$
- $1 \leq Q, P$ and $K \leq N$

Input :
4 3 2
2 1
3 1
1 4
Output :
3

Potion 1 can be brewed after potions 2 and 3 are brewed. Potion 4 can be brewed after potion 1 is brewed.
Input :
5 4 2
2 1
3 1
4 1
1 5
Output :
4