Department of Computer Science

CPS688 – Advanced Algorithms

Lab 2

General Instructions:

Due date: Week 7, two weeks from today's lab session.

Grading: Each student is required to demo the program during the lab session. Failure to show up will result in a grade of zero. No extensions.

Submission: Zip your code and submit it on D2L after your finish your demo.

Weight: 5% of your total grade.

Hint: Use code from previous labs whenever possible.

Lab Instructions:

Problem 1 – N-Queens

The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other.

The idea is to place queens one by one in different columns, starting from the leftmost column. When placing a queen in a column, you should check for clashes with already placed queens (no more than one Q in the row and column).

See the example below.

Input: your input should take the number of matrix size N. your matrix is a zeros matrix

Output: your output is a binary matrix which has 1s for the blocks where queens are placed.

Input	**Output**
4	0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0

Problem 2 – Acyclic Graph:

Given an undirected graph, you are required to check if the graph has a cycle.

Hint: consider using BFS/DFS

Input

Your program will be tested against multiple test cases. Each test case begins with two integers **n** and **e**, representing the number of **vertices** and **edges**. The next **e** lines represent the vertices that are connected by an edge.

Output

For each test case, print "no" if the graph contains a cycle; else print "yes".

Sample Input	**Sample Output**
6 6 0 1 0 3 1 2 2 4 3 4 3 5	no
6 5 0 1 0 2 0 3 1 4 2 5	yes

Problem 3 – Minimum Spanning Tree

Given a weighted undirected graph, you are required to find the minimum spanning tree in the graph using Prim's algorithm.

Input: Your program will be tested against multiple test cases. Each test case begins with two integers *n* and *e*, representing the number of vertices and edges. The next *e* lines represent the vertices that are connected by an edge and the weight of that edge.

Output: For each test case, print the sum of weights on the edges of the minimum spanning tree and the corresponding edges with their weights.

Sample Input	Sample Output
4 5 0 1 10 0 2 6 0 3 5 1 3 15 2 3 4	Edge 2-3 has a weight of 4 Edge 0-3 has a weight of 5 Edge 0-1 has a weight of 10 MST = 19

Problem 4

John enters a candy shop and in his hand is a very light-thin bag. He is given 20 mins to collect as much candy as possible, but he needs to make sure that the bag doesn't tear apart. There's a fixed number of candies in the shop – each with its own weight and sentimental value. Of course, his goal is to put as many candies in the bag as possible while also maximizing the number of best-quality candies so that he can enjoy the treat during the weekend. Plus, he doesn't want to share the candy with anyone since the candy placed in the bag is of the most value with respect to him.

But John needs your help! So, can you be kind and help him choose the candy he's required to put in the bag and the ones he's supposed to keep at the store?

Input:

- The first line contains one integer **N**; the number of candies present in the shop. Then 3 lines follow.
- The second line contains **N** integers where N_i represents the sentimental value of the candy w.r.t John
- The third line contains **N** integers where N_i represents the weight of each candy.
- The last line contains one integer **W** representing the maximum weight that can be carried by the bag

Output: Output the highest sentimental aggregated value

Sample Input	Sample Output
3 6 10 12 1 2 3 5	22

Problem 5 – Longest Increasing Subsequence

The Longest Increasing Subsequence (LIS) problem is to find the length of the longest subsequence of a given sequence such that all elements of the subsequence are sorted in increasing order. For example, the length of LIS for array A = {10, 22, 9, 33, 21, 50, 41, 60, 80} is 6 and LIS of A is {10, 22, 33, 50, 60, 80}.

Write a Dynamic programming solution for LIS. Print the LIS and the elements that formed this LIS.

Hint: this problem may be converted to a Longest Common Subsequence problem. Create another array (Array B) of unique elements of the original array (array A) and sort it. The Longest Increasing Subsequence of A must be present as a subsequence in B.

Sample Input	**Sample Output**
10 22 9 33 21 50 41 60	LIS = 5 LIS is: 10 22 33 50 60