CSE 110
A2
Bitwise Operators and Dynamic Memory Allocation

Full Marks: 20 **Time: 1 Hour**

Problem #1 (10 marks)
city distances

A two-dimensional array can be used to show distances between locations as shown in the following table.

From / To	Dhaka (Zero Point)	Bagerhat	Bandarban	Barguna	Barisal	Bhola	Bogra
Dhaka (Zero Point)	0	178	316	247	169	205	197
Bagerhat	178	0	437	160	119	155	328
Bandarban	316	437	0	469	391	427	510
Barguna	247	160	469	0	84	120	445
Barisal	169	119	391	84	0	36	366
Bhola	205	155	427	120	36	0	402
Bogra	197	328	510	445	366	402	0

Write a program that takes n number of (x, y) coordinates. Then it calculates and stores their Euclidean distances in a two dimensional array. From this distance matrix, the program finds the pair of locations which are the closest, and also the pair of locations which are the furthest from each other.

You have to use dynamic memory allocation (malloc), i.e., you are not allowed to use static array.

Sample Input(s)	**Corresponding Output(s)**
8 -1 9 -1 -1 1 1 2 0.5 2 -1 3 3 4 2 4 -7.5	1 1 2 0.5 (closest) -1 9 4 -7.5 (furthest)

Problem #2 (10 marks)

Bit rotation

Write a C program to rotate bits of a number using **bitwise operators**. The program takes an unsigned integer number u and an integer n as input. Then it performs rotation of bits of the number n times.

Rotating bits of a number to left, means shifting all bits to left and pushing the dropped Most Significant Bit (MSB) to Least Significant Bit (LSB). On the other hand, rotating bits of a number to right, means shifting all bits to right and pushing the dropped Least Significant Bit (LSB) to Most Significant Bit (MSB).

If n = +4 then the program do **right rotation** 4 times. On the other hand, if n = -4 then the program do **left rotation** 4 times.

You have to perform this operation in place, which means you can not use an additional integer or array to store the result.

Sample Input(s)	**Corresponding Output(s)**
4 9 4 2 4 -3	512 (assuming 2-byte integer) 1 32

Explanation:

0000000000000100 = 4
0000001000000000 = 512
0000000000000001 = 1
0000000000100000 = 32