COP3275 – Spring 2019

Instructor: Roozbeh Ketabi

Programming Assignment 6

1- You have to upload your codes in both Canvas and the Judge system at https://cop3275.cise.ufl.edu
2- For guide on how to access the Judge, visit Judge Access Page under Pages section in Canvas.
3- The judge will run test cases against your code and assign it a grade. You can have multiple submission for the same problem but you have to choose which one is the final (we will consider your final submission for grading). Canvas is only used for record keeping.
4- When upload on canvas, zip your programs in a zipfile with your ufid as name (nothing more, just 8 digit ufid, for instance 12345678.zip). Name your files (inside the zipped archive) p1.c, p2.c, and so on.
5- For all the problems input is read from standard input (i.e. read using scanf) and must be written to standard output (i.e. printf).
6- **Don't print extra stuff. Since the assignments are automatically graded, if the output doesn't match the expected output, you will not get the points. For instance:**
If the problem is to read a number and return its square, the expected output is a number (i.e. "printf("%d",i*i)"). If you print using something like "printf("square is %d",i*i);" you will not receive any points.
7- The assignment is due on 11:59 pm Sunday April 28th 2019. There is a 20% penalty for late submission of one day. No submission is accepted beyond that time.

Problem 1: Sparse! (4 pts.)

You are again given two vectors as input and you have to compute and print the elementwise multiplication and determine whether they are perpendicular or not (Similar to Dense! Problem). The difference is this time you are given the input in sparse representation and you have to print the computed elementwise multiplication **in sorted order of indices**. Also note that the input is given, in a **random** order (it is not sorted) so you must sort them yourself.

Sparse representation of vectors is useful when a lot of elements in the vector are zero and hence, it is wasteful to represent them using regular arrays (in terms of memory and in many cases, computation). In this format, the index and the value of each **non zero** element is given correspondingly. For instance:

V_{dense} = [1,0,0,6,0] -> V_{sparse} = [(0,1),(3,6)]

First, N the number of non zero elements of first vector is given in the first and line and then N elements are given (each in a separate line) as index<whitespace>value (e.g. "1 3"). The M (= number of nonzero elements of the second vector) and its M elements in a similar format.

Sample input (same vectors as Problem 1):	Sample output:
5 2 20 0 5 4 -30 3 10 5 5 6 6 20 0 8 2 -1 1 2 3 2 5 1	0 40 2 -20 3 20 5 5 not perpendicular

Suggestions:

- Have a struct that represents an element (let's tag it v_element) of the vector (with two integer members: index and value). Represent each vector as an array of v_element s. Sizes of the arrays are the N and M given in the input.
- Iterate over both arrays in a way that you find elements with the same index and compute the multiplication of their value. Store it in a new array.
- Use the qsort function to sort the resulting array. For that you need to write a comparator function that tells which one of the given elements is less than the other one (compare v_element indices).
- If all the elements of the results are zero, print an empty newline followed by perpendicular.

Problem 4: IP address (3 pts.)

In today's world, we are all connected through our devices. Internet Protocol (IP) is at the heart of Internet where it specifies an address for every device that is connected to the network. In general, Internet is a big network of connected subnetworks (subnets) and the addresses follow a hierarchy based on the subnets. In this problem you are given two *IP addresses* and a *Network Mask* and you have to determine whether those two IP addresses belong to the same subnet or not.

An IP address is written as four numbers between 0 and 255 separated by a dot character. For instance: 192.168.1.1 or 254.178.3.51 are valid IP addresses but 284.120.12.230 (284 is greater than 255) or 123.123..1 (third number is not given) are not. Each number can be represented with 8 bits and hence a byte (unsigned). So, we can put these 4 bytes together and store them in an integer variable that has 4 bytes (one byte per number that is separated by a dot). For instance, 254.23.10.1 can be broken down into:

254 => 11111110
23 => 00010111
10 => 00001010
1 => 00000001

So, the corresponding 32 bit integer is 11111110000101110000101000000001 which when seen as an unsigned integer in decimal (base 10) is 4,262,922,753.

A network mask is a number that is used to mask the exact address of the device and instead show the subnet address. Typically, because of the Internet's hierarchy, this number when represented in binary, has its N most significant bits set to 1. For instance, 11111111100000000000000000000000 (or in decimal 4,286,578,688) is a valid subnet mask (note that because of equivalence of IP address in their dot notation to integer numbers we can also write a subnet mask in the IP format. For instance, the subnet mask above can be written as 255.128.0.0 - but you don't need this conversion for this problem).

When you ***bitwise and*** the IP address and the subnet mask, it extracts the most significant bits of the given IP address. It is usually known as the subnet address. Two IPs belong to the same subnet if they have the same subnet address.

In this problem, you are given two IP addresses in dot notation and a number N corresponding to the number of bits that are set to 1 in the subnet mask. Print 1 if they belong to the same subnet and 0 otherwise.

Sample input:	**Sample output:**
192.168.1.10 192.172.23.144 13	1
10.10.20.20 11.10.20.20 7	1
10.10.20.20 11.10.20.20 8	0