DSA Fall 2024
Exam,1st Sitting
Problem 1

Problem Requirements:

Write the necessary data structures and algorithm to find the network address to which a device with a particular IP is connected.

An IP address is a 32 bit unsigned integer. It is usual that it is presented in human readable format where it is represented as 4 bytes separated by a dot (.) .

Ex: 0xC1E20C02 will be 193.226.12.2 in human readable format.

A network mask is 4 bytes unsigned integer similar to the IP address. In human readable format a network mask would look like, for example: 255.255.255.0

A network address can be found by performing a bitwise AND operation between the IP address and the network mask.

For the purposes of this problem we will not care for any other real life constraints for these concepts not mentioned above.

Your program will read as command line arguments the IP address in hex format, with capital letters, and a network mask in human readable format and prints to the standard output the network address associated with those IP and network mask in human readable format. Take great care that there are no leading or trailing whitespaces for the output of your code.

your_program   IP_in_hex   Network_mask_in_human_readeable_format

Ex1: For the call

name_of_your_program 0xC1E20C02 255.255.255.0

the output will be:

193.226.12.0

Ex2: For the call:

name_of_your_program 0xAC1EFF04 255.255.248.0

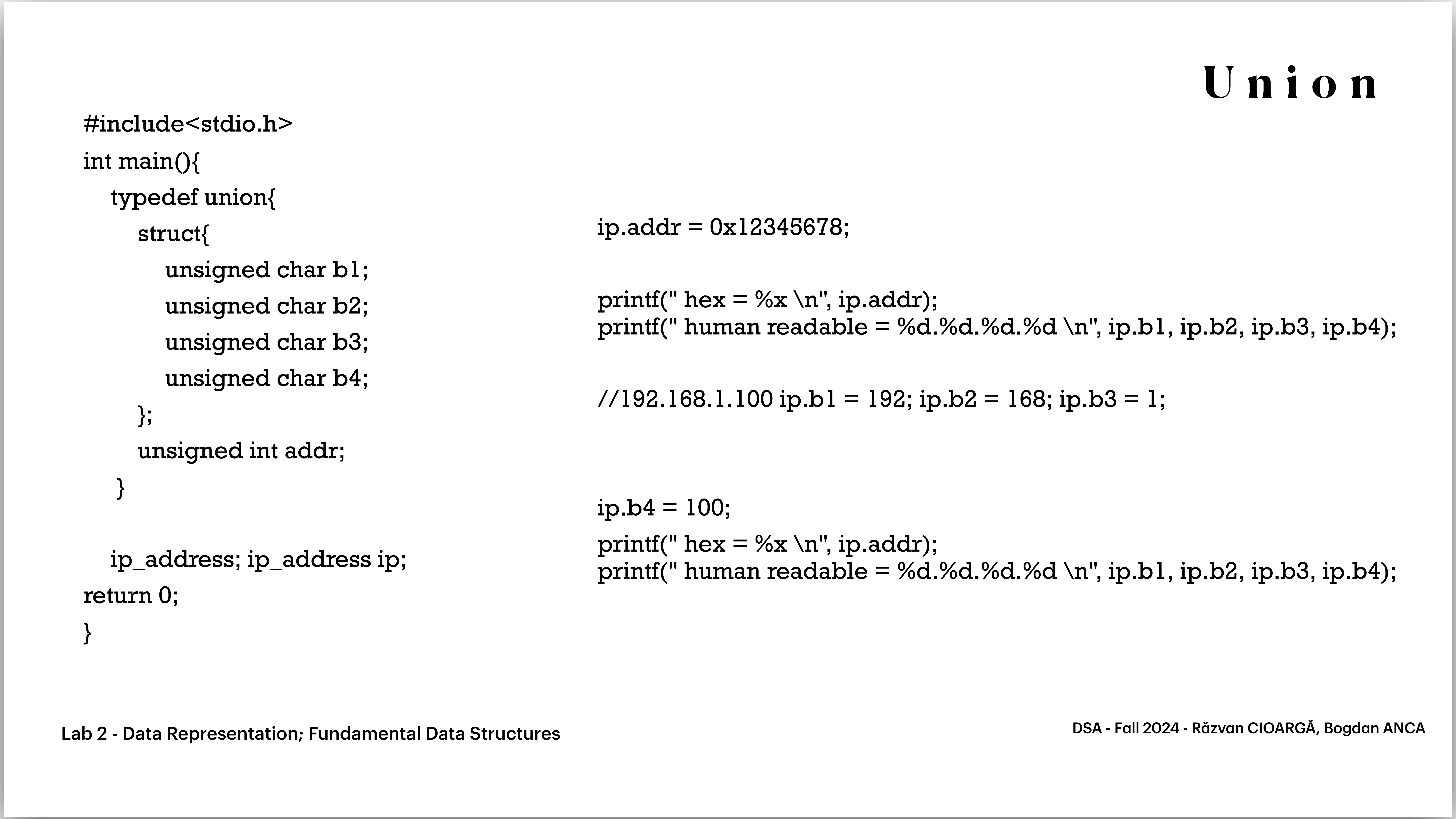
the output will be:

172.30.248.0

Points will be deducted for unnecessary code.

Your solution will be one .c file with the following naming scheme: **surname_name_s2.c**. If your name has special characters, find the best english alphabet substitution, if that substitution cannot be found (such for example for an apostrophe) omit that character.

I wish to bring to your notice slide 14 from Week 1 Lab, which contains code written and explained during the lab.
The slide was available for study on the courses page on cv.upt.ro
Same example was given regarding the union data type during Lecture 1.
The union data structure as implemented in the C programming language was, most probable, a topic as well in Computer Programming Course 1st Year Semester 1



Union

```
#include<stdio.h>
int main(){
    typedef union{
        struct{
            unsigned char b1;
            unsigned char b2;
            unsigned char b3;
            unsigned char b4;
        };
        unsigned int addr;
     }

    ip_address; ip_address ip;
return 0;
}
```

```
ip.addr = 0x12345678;


printf(" hex = %x \n", ip.addr);
printf(" human readable = %d.%d.%d.%d \n", ip.b1, ip.b2, ip.b3, ip.b4);


//192.168.1.100 ip.b1 = 192; ip.b2 = 168; ip.b3 = 1;



ip.b4 = 100;
printf(" hex = %x \n", ip.addr);
printf(" human readable = %d.%d.%d.%d \n", ip.b1, ip.b2, ip.b3, ip.b4);
```

Lab 2 - Data Representation; Fundamental Data Structures

DSA - Fall 2024 - Răzvan CIOARGĂ, Bogdan ANCA

Solution:

We are going to use the union data structure for data representation to solve the problem because we can take advantage of the automatic data conversion between one 32 bits unsigned integer and four 8 bit unsigned integers. Both the hex representation and the human readable representation are stored in the same location in memory (exact same bits), the only difference is the way we want that data to be formatted.

```c
#include<stdio.h>
#include<stdint.h>

typedef union{
  uint32_t ip_addr;
  struct {
    uint8_t b1;
    uint8_t b2;
    uint8_t b3;
    uint8_t b4;
  };

} IP;


int main(int argc, char **argv){


  IP addr = {0};
  IP mask = {0};
  IP net  = {0};

  sscanf(argv[1], "%X", &addr.ip_addr);
  sscanf(argv[2], "%hhu.%hhu.%hhu.%hhu", &mask.b4, &mask.b3, &mask.b2, &mask.b1);

  net.ip_addr = addr.ip_addr & mask.ip_addr;

  printf("%d.%d.%d.%d", net.b4, net.b3, net.b2, net.b1);
return 0;
}
```

Equivalent to

```c
typedef union{
  unsigned int ip_addr;
  struct {
    unsigned char b1;
    unsigned char b2;
    unsigned char b3;
    unsigned char b4;
  };

}
```

We can use **uint32_t** and **uint8_t** datatypes instead of **unsigned int** and **unsigned char**.

In **sscanf** and **printf** the order of the uint8 fields is reversed from datatype definition. This is because of the endianness of the processor (discussed in the Week 1 laboratory, and more in depth in Computer Architecture course). This was not something for which points were lost for the solution, but an attentive engineer student would have figured out that the printout was in reversed order.

Error checking was omitted for brevity. There were no tests where the arguments were missing or were not formatted correctly.

Declaring and initialising variables

Read the values from the command line arguments in their specified format
%X formats a hex represented integer with capital letters
%hhu formats a short short unsigned int (8 bit)

"The Algorithm" : bitwise AND between the two variables

Print human readable format of network mask