Adresarea IPv4

Responsabili:

- Dragos Unguru
- Vlad Burcea
- Mihai Constantin
- Stefan Vodita

Termen de predare: **28.10.2019, ora 23:55**
Pentru fiecare zi (24 de ore) de întârziere, se vor scădea 10 puncte din nota acordată. Temele trimise după 7 de zile de întârziere vor putea fi notate cu maxim 30 de puncte. În consecință, deadline-ul hard este **04.11.2019, ora 23:55**.

Întrebări

Dacă aveți nelămuriri, puteți să ne contactați pe forumul dedicat temei de casă nr. 1. La orice întrebare vom răspunde în maxim 24 de ore.
Nu se acceptă întrebări în ultimele 24 de ore înainte de deadline.

Actualizări:

- **[25.10.2019 - 10:30]** adaugat fisier sursa in arhiva care contine o functie pentru afisarea primului octet dintr-un unsigned int

Obiective Temă

- să se realizeze un program urmând anumite cerințe
- folosirea operatorilor pe biți
- să se respecte formate stricte de intrare/ieşire
- să se însușească cunoștințele din primele trei laboratoare

Cerință

În această temă o să lucrați cu cu adrese IPv4 adresele IP. Programul va citi de la tastatură o serie de date independente (câte o serie pe linie) pe care va trebui să le prelucreze.

Adresarea IP

Ne propunem să sistematizăm noțiunile de adresă IP, mască de rețea, adresă de rețea și adresă de broadcast.

Adresa IP asigură conectivitatea echipamentelor (calculatoare, telefoane, rutere, servere, etc.) în rețea/Internet.

În cazul unei adrese IP, vom configura, tot timpul, următoarele:

- **adresa IP** - 4 grupuri a câte 8 biți. Exemplu: `192.168.100.200`[1)]

- **masca de rețea** (subnet mask) - 4 grupuri a câte 8 biți, cu proprietatea că se începe cu bitul 1, iar toți biții de 1 sunt consecutivi, alternanța 0/1 fiind interzisă. De exemplu `11111111.00000000.00000000.00000000` este o mască de rețea validă, iar `11000001.00000000.00000000.0000000` este o mască nevalidă. Pentru a ușura citirea măștii acestea se scrie în zecimal, similar adresei IP: `11111111.00000000.00000000.00000000` = `255.0.0.0`. Datorită proprietății speciale în care biții de 1 sunt consecutivi o altă formă în care veți mai găsi specificată masca de rețea este forma prefixată: `/X`, unde X reprezintă numărul de biți de 1: `11111111.00000000.00000000.00000000` = `255.0.0.0` = `/8`.

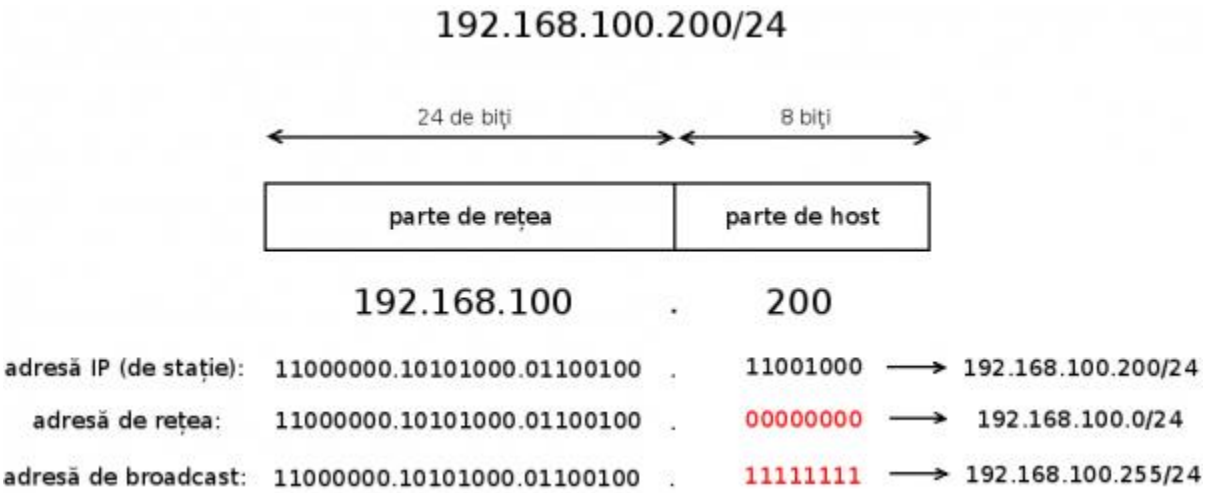
Pornind de la adresa IP și masca de rețea putem identifica două alte proprietăți ale unei rețele (pentru exemplificare vom folosi adresa IP 192.168.100.200/255.255.255.0):

- **adresa de rețea** - se obține făcând **ȘI-logic** între biții adresei IP și biții măștii de rețea
  - `192.168.100.200 & 255.255.255.0 =` **`192.168.100.0`**
- **adresa de broadcast** - se obține făcând **SAU-logic** între biții adresei IP și biții din complementul măștii de rețea (complementul se obține inversând valoarea biților de pe fiecare poziție)
  - `192.168.100.200 | 0.0.0.255 =` **`192.168.100.255`**

Atunci când cunoaștem adresa IP și masca de rețea și vrem să obținem adresa de rețea și adresa de broadcast, este util să folosim masca de rețea pentru a împărți adresa IP în două:

- O **parte de subrețea**, care se întinde pe câți biți de 1 are masca de rețea. E vorba de 24 de biți, pentru o mască /24 (sau 255.255.255.0) sau 16 biți pentru o mască /16 (sau 255.255.0.0) sau 20 de biți pentru o mască /20 (sau 255.255.240.0).
- O **parte de stație** (sau parte de host) care se întinde pe restul spațiului (32 minus numărul de biți de 1 ai măștii de rețea). E vorba de 8 biți pentru o mască /24 (32-24 = 8) sau de 16 biți pentru o mască /16 (32-16 = 16) sau de 12 biți pentru o mască /20 (32-20 = 12).

Pe această împărțire pentru adresa 192.168.100.200/24 vom obține aceleași valori precum cele calculate mai sus, lucru reflectat și în figura de mai jos.



Să obținem adresa de rețea și adresa de broadcast pentru adresa `172.16.200.100/20`.
Transformăm adresa într-o adresă hibridă punând biți de 0 acolo unde se găsește masca de rețea: al treilea octet din cei patru ai adresei IP: `172.16.1100|1000.xxxxxxxx`. Am folosit operatorul | (pipe) pentru a separa **partea de rețea** (primii 20 de biți, aferenți rețelei) de **partea de stație (*host*)** (ceilalți biți (32-20 = 12 biți) aferenți stației). Nu sunt relevanți pentru calculul nostru biții ultimului octet așa că am pus `xxxxxxxx` în locul lor.
Adresa de rețea are **toți biții de stație puși pe 0**, deci va fi `172.16.1100|0000.00000000`. Rezultă adresa de rețea `172.16.192.0/20`.
Adresa de broadcast are **toți biții de stație puși pe 1**, deci va fi `172.16.1100|1111.11111111`. Rezultă adresa de broadcast `172.16.207.255/20`.
Considerăm că adresele IP pot lua orice valori.

Structura unui set de date

Un set de date (o linie) va fi compus din mai multe elemente. Elementele vor fi întotdeauna separate printr-un spațiu.

Unele elemente sunt utile doar pentru implementarea anumitor task-uri. Dacă nu doriți să implementați acele task-uri, va trebui să găsiți o metodă prin care să le ignorați (ex: să le citiți, dar să nu faceți neapărat ceva util cu ele).

Format:

```
MSK_1 MSK_2 IP_1 IP_2 N [NET_1 NET_2 ... NET_N]
```

- **MSK_1**: o masca de rețea
- **MSK_2**: numarul de biti setati din masca de rețea pentru IP_1 si IP_2
- **IP_1**: o adresă IP
- **IP_2**: o altă adresă IP
- **N**: număr întreg natural
- **[NET_1 ... NET_N]**: listă de N adrese de rețea

Structura outputului

Pe prima linie se va citi numărul de seturi de date. Veți avea de rezolvat mai multe taskuri. Puteți să implementați doar o parte din ele. Pentru fiecare set de date se va afișa output-ul corespunzător tuturor task-urilor implementate, ulterior se va trece la un alt set de date, dacă este cazul. Fiecare set de date va fi precedat de o linie cu numărul setului de date. Astfel, dacă cineva a rezolvat doar task-ul 0 și task-ul 2, iar testul conține 2 seturi de date:

- pe prima linie se va afișa: 1
- pe a doua linie se va afișa output-ul corespunzător task-ului 0 al primului set de date precedat de -0
- pe a treia linie se va afișa output-ul corespunzător task-ului 2 al primului set de date precedat de -2
- pe a patra linie se va afișa: 2
- pe a cincea linie se va afișa output-ul corespunzător task-ului 0 al celui de-al doilea set de date precedat de -0
- pe a șasea linie se va afișa output-ul corespunzător task-ului 2 al celui de-al doilea set de date precedat de -2

Task 0 (5p)

Afișați adresa IP_1 împreună cu MSK_2 în formatul: IP_1/MSK_2.

Exemplu

Input:

```
1

255.237.0.0  20  192.168.25.87  192.168.26.1  3  192.168.0.0/16  192.0.26.0/16
192.168.26.0/24
```

Output:

```
-0 192.168.25.87/20
```

Task 1 (5p)

Afișați masca MSK_2 în format zecimal.

Exemplu

Input:

```
1

255.237.0.0  20  192.168.25.87  192.168.26.1  3  192.168.0.0/16  192.0.26.0/16
192.168.26.0/24
```

Output:

```
-1 255.255.240.0
```

Task 2 (5p)

Afișați valorile: MSK_1 în baza 8 și în baza 16, separate printr-un spațiu. Nu este necesar să faceți conversia explicită. Puteți să vă folosiți de facilitățile oferite de printf.

Exemplu

Input:

```
1

255.237.0.0  20  192.168.25.87  192.168.26.1  3  192.168.0.0/16  192.0.26.0/16
192.168.26.0/24
```

Output:

```
-2 377.355.0.0 FF.ED.0.0
```

Începând cu acest task vom lucra doar pe rețeaua a cărei mască este MSK_2, dacă nu se specifică altfel.

Task 3 (5p)

Afișați adresa de rețea a lui IP_1.

Exemplu

Input:

```
1

255.237.0.0  20  192.168.25.87  192.168.26.1  3  192.168.0.0/16  192.0.26.0/16
192.168.26.0/24
```

Output:

```
-3 192.168.16.0
```

Task 4 (5p)

Afișați adresa de broadcast a lui IP_1.

Exemplu

Input:

```
1
```

```
255.237.0.0  20  192.168.25.87  192.168.26.1  3  192.168.0.0/16  192.0.26.0/16
192.168.26.0/24
```

Output:

```
-4 192.168.31.255
```

Task 5 (10p)

Determinați dacă IP_2 face parte din aceeași rețea cu IP_1. Dacă răspunsul este afirmativ, atunci se va afișa: DA, altfel NU.

Exemplu

Input:

```
1

255.237.0.0  20  192.168.25.87  192.168.26.1  3  192.168.0.0/16  192.0.26.0/16
192.168.26.0/24
```

Output:

```
-5 DA
```

Task 6 (15p)

Verificați corectitudinea măstii de rețea MSK_1. Dacă răspunsul este afirmativ, atunci se va afișa: DA, altfel NU.

Exemplu

Input:

```
1

255.237.0.0  20  192.168.25.87  192.168.26.1  3  192.168.0.0/16  192.0.26.0/16
192.168.26.0/24
```

Output:

```
-6 NU
```

Task 7 (20p)

Dacă răspunsul la task-ul 6 a fost afirmativ, atunci afișați MSK_1. Altfel, corectați masca MSK_1 setând pe 0 toți biții care succed cel mai semnificativ bit setat pe 0 și afișați după corectare acea mască.

Note: Atât pentru acest task cât și pentru cel precedent nu se vor lua în considerare soluțiile care doar afișeaza "DA" sau "NU", respectiv MSK_1, chiar dacă vor trece anumite teste pe checker.

Exemplu

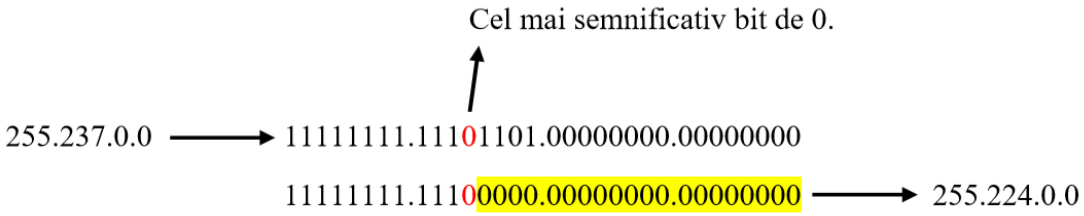
Input:

```
1

255.237.0.0  20  192.168.25.87  192.168.26.1  3  192.168.0.0/16  192.0.26.0/16
192.168.26.0/24
```

Output:

```
-7 255.224.0.0
```



Explicație

Task 8 (10p)

Afișați IP_1 reprezentând în binar fiecare octet.

Exemplu

Input:

```
1

255.237.0.0  20  192.168.25.87  192.168.26.1  3  192.168.0.0/16  192.0.26.0/16
192.168.26.0/24
```

Output:

```
-8 11000000.10101000.00011001.01010111
```

Task 9 (20p)

Afișați pe un rand indicii rețelelor din care ar putea face parte IP_2, separate de cate un spațiu. (Indexarea începe de la 0).

Exemplu

Input:

```
1
```

```
255.237.0.0  20  192.168.25.87  192.168.26.1  3  192.168.0.0/16  192.0.26.0/16
192.168.26.0/24
```

Output:

```
-9 0 2
```

Exemplu complet input/output

Input:

```
1

255.237.0.0  20  192.168.25.87  192.168.26.1  3  192.168.0.0/16  192.0.26.0/16
192.168.26.0/24
```

Output:

```
1

-0 192.168.25.87/20

-1 255.255.240.0

-2 377.355.0.0 FF.ED.0.0

-3 192.168.16.0

-4 192.168.31.255

-5 DA

-6 NU

-7 255.224.0.0

-8 11000000.10101000.00011001.01010111

-9 0 2
```

Deoarece citirea se face de la tastatură, iar afișarea se face pe ecran, pentru a putea testa mai ușor puteți folosi redirectări de genul:

```
./ip < input.txt > output.txt
```

Trimitere temă

Tema va fi trimisă folosind vmchecker, cursul **Programarea Calculatoarelor (CB & CD)**. Găsiți arhiva cu checker-ul si makefile-ul aici. Instrucțiunile de rulare ale checkerului le găsiti in fișierul "checker_README" din arhiva

După cum probabil ați observat, task-urile au un total de **100 de puncte**, dar motivul pentru acest lucru este ca fiecare test să valoreze **1 punct** (*adica 0.01 puncte din nota finală*). Dacă soluția voastră trece toate testele veți acumula un total de **0.9 puncte** (din nota finală). Celelalte **0.1 puncte** se vor acorda pentru **coding style** si **README**.
Formatul arhivei va fi următorul:

1. fișierul `ip.c`.
2. Fișierul **Makefile** dat de noi sau unul făcut de voi (detalii aici) care să conțină următoarele reguli:
    a. **build**: creează executabilul aferent (numele executabilului: **ip**)
    b. **run**: rulează executabilul aferent
    c. **clean**: șterge fișierele obiect/executabile create.
3. Un fișier README în care vă descrieți rezolvarea fiecărui task.

1. Arhiva trebuie să fie de tipul **zip**.
2. Inputul se va fi citit de la **stdin (tastatura)**, iar output-ul va fi afișat la **stdout (ecran)**. Testarea se face cu ajutorul redirectării acestora din linia de comandă/bash.

Listă depunctări

Lista nu este exhaustivă.

- o temă care nu compilează și nu a rulat pe **vmchecker** nu va fi luată în considerare
- o temă care nu rezolvă cerința și trece testele prin alte mijloace nu va fi luată în considerare
- [-1.0]: numele variabilelor nu sunt sugestive
- [-1.0]: linii mai lungi de 80 de caractere
- [-5.0]: abordare ineficientă
    - în cadrul cursului de programare nu avem ca obiectiv rezolvarea în cel mai eficient mod posibil a programelor; totuși, ne dorim ca abordarea să nu fie una ineficientă, de genul să nu folosiți instrucțiuni repetitive acolo unde clar nu era cazul, etc.

[1)] valoarea maximă pentru fiecare grup este 255 = 2^8-1