

Il codice sotto riportato rappresenta un'applicazione semplificata per il calcolo dell'area di alcuni selezionati poligoni. Un utente valuta se nella lista di poligoni supportati dall'applicazione c'è quello di cui vuole calcolare l'area. In caso affermativo, specifica i dati necessari per il calcolo dell'area e ottiene la soluzione dal server. Si risponda ai seguenti quesiti:

Q1 Completare il codice mancante nel Server e nel Client in maniera tale che :

1. Il server sia in ascolto sulla porta 13001
2. Il servizio di trasporto utilizzato sia TCP

Q2 Si riporti ciò che appare sul terminale client nella situazione in cui il client voglia calcolare l'area di un Triangolo di base = 2 [m] e altezza = 1[m]

Q1) Vedere sotto

Q2)

Ciao!

Sono supportati i seguenti Poligoni:

"['Triangolo', 'Rettangolo']"

Vuoi Calcolare l'area [si/no]? si

Scegli un poligono nella lista proposta: Triangolo

Inserisci un valore numerico per la base (U. di misura: metro): 2

Inserisci un valore numerico per l'altezza (U. di misura: metro): 1

L'area del poligono considerato è: 1.0 m²

#Script client

```
from socket import *
```

```
serverName = 'localhost'
```

```
serverPort = 13001
```

```
ClientSock = socket(AF_INET, SOCK_STREAM)
```

```
ClientSock.connect((serverName, serverPort))
```

```
print('Ciao!')
```

```
pol_disp = ClientSock.recv(100)
```

```
print('Sono supportati i seguenti Poligoni: ')
```

```
print(pol_disp)
```

```
decisione = input('Vuoi Calcolare l\'area [si/no]? ')
```

```
if decisione == 'si':
```

```
    # Selezione una delle scelte disponibili
```

```
    poligono = input('Scegli un poligono nella lista proposta: ')
```

```
    ClientSock.send(poligono.encode('utf-8'))
```

```
    flag = ClientSock.recv(100).decode('utf-8')
```

```
    if flag == 'ok':
```

```
        base = input('Inserisci un valore numerico per la base (U. di misura: metro): ')
```

```
        ClientSock.send(base.encode('utf-8'))
```

```
        altezza = input('Inserisci un valore numerico per l\'altezza (U. di misura: metro): ')
```

```
        ClientSock.send(altezza.encode('utf-8'))
```

```
        risposta = ClientSock.recv(100).decode('utf-8')
```

```
        if risposta == 'ok':
```

```
            area = ClientSock.recv(100).decode('utf-8')
```

```
            print('L\'area del poligono considerato è: '+str(area)+' m2')
```

```
        else:
```

```
            print('Base o Altezza sono errate (devono essere numeri). Chiusura Sessione')
```

```
    else:
```

```

        print('Soluzione non disponibile, chiusura sessione.')
    elif decisione == 'no':
        ClientSock.send('no'.encode('utf-8'))
        print('Arrivederci!')
    else:
        print('Scelta errata, chiusura sessione!')
    ClientSock.close()

```

#Script server

```

from socket import *

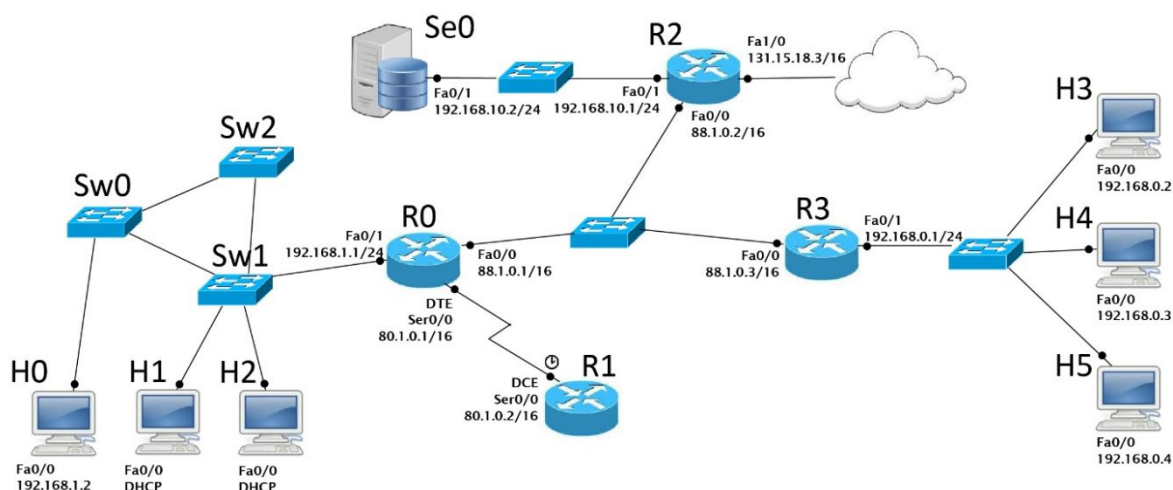
def lista_poligoni():
    return ['Triangolo', 'Rettangolo']
def area_triangolo(base, altezza):
    return base*altezza*0.5
def area Rettangolo(base, altezza):
    return base*altezza

serverPort = 13001
serverSock = socket(AF_INET, SOCK_STREAM)
serverSock.bind('', serverPort)
serverSock.listen(1)
print('Il server è pronto a ricevere')

while True:
    clSock, clAddr= serverSock.accept()
    print("Connection from: ", clAddr)
    # Invia Al Client i Poligoni disponibili
    poligons = str(lista_poligoni())
    clSock.send(poligons.encode('utf-8'))
    selezione = clSock.recv(1024).decode('utf-8')
    if str(selezione) in poligons:
        clSock.send('ok'.encode('utf-8'))
        base = clSock.recv(1024).decode('utf-8')
        altezza = clSock.recv(1024).decode('utf-8')
        try:
            base = float(base)
            altezza = float(altezza)
            clSock.send('ok'.encode('utf-8'))
            if str(selezione) in ['Triangolo']:
                area = area_triangolo(base,altezza)
            elif str(selezione) in ['Rettangolo']:
                area = area Rettangolo(base, altezza)
            clSock.send(str(area).encode('utf-8'))
        except:
            clSock.send('Errore'.encode('utf-8'))
    elif str(selezione) == 'no':
        print("")
    else:
        clSock.send('Errore'.encode('utf-8'))
    clSock.close()

```

Si consideri la rete in figura



Attenzione:

- Indirizzi IP e gateway sono già stati configurati per i 6 host.
- Le interfacce dei router R0, R1 e R2 sono già state configurate ed attivate come in figura.
- Le reti /24 sono reti private
- Indicare sempre prima del comando il prompt visualizzato dal sistema, prestando attenzione alla modalità di partenza in ciascuna richiesta

Q1) Configurare ed attivare entrambe le interfacce del router **R3**. (1 punto)

Per l'interfaccia Fa0/0:

R3> enable

R3# configure terminal

R3(config)# interface Fa0/0

R3(config-if)# ip address 88.1.0.3 255.255.0.0

R1(config-if)# no shutdown

Q2) Abilitare RIPv1 sul router **R2** in modo tale che vengano dichiarate solo le reti pubbliche. Una delle seguenti configurazioni è corretta mentre le altre due contengono errori. Rispondi con la lettera della corretta configurazione (1 punto):

R2>enable

R2#configure terminal

R2(config)#router rip

R2(config-router)#version 1

R2(config-router)#network 88.1.0.0

R2(config-router)#network 131.15.0.0

Q3) Configurare il port forwarding sul router **R2** in modo che sia effettuato il seguente mapping:

IP	Port	IP	Port
88.1.0.2	18120	192.168.10.2	12000

(1 punto):

R2>enable

R2#configure terminal

R2(config)#interface FastEthernet 0/1

R2(config-if)#ip nat inside

R2(config-if)#exit

R2(config)#interface FastEthernet 0/0

R2(config-if)#ip nat outside

R2(config-if)#exit

R2(config)#ip nat inside source static tcp 192.168.10.2 12000 88.1.0.2 18120