```python
#pagella client per pagellaServerMulti_V01.py versione 1.0 multithread
#nome del file : pagellaClientMulti_V01.py
import socket
import sys
import random
import os
import time
import threading
import multiprocessing
import json
import pprint

SERVER_ADDRESS = '127.0.0.1'
SERVER_PORT = 22001
NUM_WORKERS=1

def genera_richieste1(num,address,port):
    #start_time_thread= time.time()
    try:
        s=socket.socket()
        s.connect((address,port))
        print(f"\n{threading.current_thread().name} {num+1}) Connessione al server: {address}:{port}")
    except:
        print(f"{threading.current_thread().name} Qualcosa è andato storto, sto uscendo... \n")
        sys.exit()

    #1. Generazione casuale:
    #   di uno studente(valori ammessi: 5 cognomi a caso scelti da una lista)
    #   di una materia(valori ammessi: Matematica, Italiano, inglese, Storia e Geografia)
    #   di un voto (valori ammessi 1 ..10)
    #   delle assenze (valori ammessi 1..5)
    studenti=['Studente0','Studente1','Studente2','Studente3','Studente4']
    materie=['Matematica','Italiano','inglese','Storia e Geografia']
    voto=random.randint(1,10)
    assenze=random.randint(1,5)
    materia=materie[random.randint(0,3)]
    studente=studenti[random.randint(0,4)]

    #2. comporre il messaggio, inviarlo come json e ricevere il risultato
    messaggio={'studente':studente,
    'materia':materia,
    'voto':voto,
    'assenze':assenze}
    print(f"Dati inviati al server {messaggio}")
    messaggio=json.dumps(messaggio)
    s.sendall(messaggio.encode("UTF-8"))
    data=s.recv(1024)
    data=data.decode()
    data=json.loads(data)
    print(f"Dati ricevuti dal server {data}")

    if not data:
        print(f"{threading.current_thread().name}: Server non risponde. Exit")
    else:
        # completare:
        #1. recuperare studente, materia, voto e assenze
        studente=data['studente']
        materia=data['materia']
        print(f"{threading.current_thread().name}: La valutazione di {data['studente']} in {data['materia']} è {data['valutazione']} ")
    s.close()
    #end_time_thread=time.time()
    #print(f"{threading.current_thread().name} tempo di esecuzione time=", end_time_thread-start_time_thread)
```

```python
def genera_richieste2(num,address,port):
    #start_time_thread= time.time()
    try:
        s=socket.socket()
        s.connect((address,port))
        print(f"\n{threading.current_thread().name} {num+1}) Connessione al server: {address}:{port}")
    except:
        print(f"{threading.current_thread().name} Qualcosa è andato storto, sto uscendo... \n")
        sys.exit()

    #1. Generazione casuale di uno studente(valori ammessi: 5 cognomi a caso scelti da una lista)
    #   Per ognuna delle materie ammesse: Matematica, Italiano, inglese, Storia e Geografia)
    #   generazione di un voto (valori ammessi 1 ..10)
    #   e delle assenze (valori ammessi 1..5)
    #   esempio: pagella={'studente': 'Studente2', 'pagella': [('Matematica', 1, 1), ('Italiano', 3, 3), ('inglese', 5, 4), ('Storia e Geografia', 1, 1)]}

    studenti=['Studente0','Studente1','Studente2','Studente3','Studente4']
    materie=['Matematica','Italiano','inglese','Storia e Geografia']
    studente=studenti[random.randint(0,4)]
    pagella=[]
    for m in materie:
        voto=random.randint(1,10)
        assenze=random.randint(1,5)
        pagella.append((m,voto,assenze))

    #2. comporre il messaggio, inviarlo come json e ricevere il risultato
    messaggio={'studente':studente,
    'pagella':pagella}
    print(f"Dati inviati al server {messaggio}")
    messaggio=json.dumps(messaggio)
    s.sendall(messaggio.encode("UTF-8"))
    data=s.recv(1024)
    data=data.decode()
    data=json.loads(data)
    print(f"Dati ricevuti dal server {data}")

    if not data:
        print(f"{threading.current_thread().name}: Server non risponde. Exit")
    else:
        # completare:
        #1. recuperare studente, media voti e totale assenze

        print(f"{threading.current_thread().name}: Lo studente {data['studente']} ha una media di: {data['media']:.2f} e un totale di assenze : {data['assenze']} ")
    s.close()
    #end_time_thread=time.time()
    #print(f"{threading.current_thread().name} tempo di esecuzione time=", end_time_thread-start_time_thread)
```

```python
def genera_richieste3(num,address,port):
    #start_time_thread= time.time()
    try:
        s=socket.socket()
        s.connect((address,port))
        print(f"\n{threading.current_thread().name} {num+1}) Connessione al server: {address}:{port}")
    except:
        print(f"{threading.current_thread().name} Qualcosa è andato storto, sto uscendo... \n")
        sys.exit()

    #  1. Per ognuno degli studenti ammessi: 5 cognomi a caso scelti da una lista
    #   Per ognuna delle materie ammesse: Matematica, Italiano, inglese, Storia e Geografia)
    #   generazione di un voto (valori ammessi 1 ..10)
    #   e delle assenze (valori ammessi 1..5)
    #   esempio: tabellone={"Cognome1":[("Matematica",8,1), ("Italiano",6,1), ("Inglese",9,3), ("Storia",8,2), ("Geografia",8,1)],
    #                       "Cognome2":[("Matematica",7,2), ("Italiano",5,3), ("Inglese",4,12), ("Storia",5,2), ("Geografia",4,1)],
    #                       .....}
    #2. comporre il messaggio, inviarlo come json

    studenti=['Studente0','Studente1','Studente2','Studente3','Studente4']
    materie=['Matematica','Italiano','inglese','Storia e Geografia']
    tabellone={}
    for stud in studenti:
        pagella=[]
        for m in materie:
            voto=random.randint(1,10)
            assenze=random.randint(1,5)
            pagella.append((m,voto,assenze))
        tabellone[stud]=pagella
    #2. comporre il messaggio, inviarlo come json e ricevere il risultato
    #messaggio={'studente':studente,
    #'pagella':pagella}
    print("Dati inviati al server")
    pp=pprint.PrettyPrinter(indent=4)
    pp.pprint(tabellone)
    tabellone=json.dumps(tabellone)
    s.sendall(tabellone.encode("UTF-8"))
    data=s.recv(1024)
    data=data.decode()
    data=json.loads(data)
    print("Dati ricevuti dal server")
    pp.pprint(data)

    if not data:
        print(f"{threading.current_thread().name}: Server non risponde. Exit")
    else:
        # completare:
        #1. recuperare il tabellone e stampare i dettagli di ogni singolo studente
        for elemento in data:
            print(f"{threading.current_thread().name}: Lo studente {elemento['studente']} ha una media di: {elemento['media']:.2f} e un totale di assenze : {elemento['a
    s.close()
    #end_time_thread=time.time()
    #print(f"{threading.current_thread().name} tempo di esecuzione time=", end_time_thread-start_time_thread)
```

```python
if __name__ == '__main__':
    start_time=time.time()
    # PUNTO A) ciclo per chiamare NUM_WORKERS volte la funzione genera richieste (1,2,3)
    # alla quale passo i parametri (num,SERVER_ADDRESS, SERVER_PORT)
    for num in range (0,NUM_WORKERS):
        genera_richieste1(num,SERVER_ADDRESS, SERVER_PORT)
        #genera_richieste2(num,SERVER_ADDRESS, SERVER_PORT)
        #genera_richieste3(num,SERVER_ADDRESS, SERVER_PORT)
    end_time=time.time()
    print("Total SERIAL time=", end_time - start_time)

    start_time=time.time()
    threads=[]
    # PUNTO B) ciclo per chiamare NUM_WORKERS volte la funzione genera richieste (1,2,3)
    # tramite l'avvio di un thread al quale passo i parametri args=(num,SERVER_ADDRESS, SERVER_PORT,)
    # avviare tutti i thread e attenderne la fine
    for num in range(NUM_WORKERS):
        threads.append(threading.Thread(target=genera_richieste1, args=(num,SERVER_ADDRESS, SERVER_PORT,)))
        #threads.append(threading.Thread(target=genera_richieste2, args=(num,SERVER_ADDRESS, SERVER_PORT,)))
        #threads.append(threading.Thread(target=genera_richieste3, args=(num,SERVER_ADDRESS, SERVER_PORT,)))
    [thread.start() for thread in threads]
    [thread.join() for thread in threads]
    end_time=time.time()
    print("Total THREADS time= ", end_time - start_time)

    start_time=time.time()
    process=[]
    # PUNTO C) ciclo per chiamare NUM_WORKERS volte la funzione genera richieste (1,2,3)
    # tramite l'avvio di un processo al quale passo i parametri args=(num,SERVER_ADDRESS, SERVER_PORT,)
    # avviare tutti i processi e attenderne la fine
    for num in range(NUM_WORKERS):
        process.append(multiprocessing.Process(target=genera_richieste1, args=(num,SERVER_ADDRESS, SERVER_PORT,)))
        #process.append(multiprocessing.Process(target=genera_richieste2, args=(num,SERVER_ADDRESS, SERVER_PORT,)))
        #process.append(multiprocessing.Process(target=genera_richieste3, args=(num,SERVER_ADDRESS, SERVER_PORT,)))
    [process.start() for process in process]
    [process.join() for process in process]
    end_time=time.time()
    print("Total PROCESS time= ", end_time - start_time)
```