```
"""
Client program for communication via UDP sockets. Takes a port, host and request
type. Sends the server a request for either date or time, and then receives a
response packet. Prints the payload  containing the desired information.
Author: Vikas Shenoy vsh33
Date: 16/8/2018
"""
import socket
import select
from sys import argv



def isValidResponse(packet):
    """Checks the response packet, ensuring the packet has the correct size
    and header. Takes the response packet. Returns a boolean of whether the
    packet is valid or not."""
    valid = True
    magicNumber = packet[0] << 8 | packet[1]
    packetType = packet[2] << 8 | packet[3]
    languageCode = packet[4] << 8 | packet[5]
    year = packet[6] << 8 | packet[7]
    month = packet[8]
    day = packet[9]
    hour = packet[10]
    minute = packet[11]
    length = packet[12]

    # Error checking for response packet
    if len(packet) < 13:
        print("Response packet must be at least 13 bytes long")
        valid = False
    elif magicNumber != 0x497E:
        print("Response pkt must have magic no 0x497E")
        valid = False
    elif packetType != 0x0002:
        print("Response pkt must have pkt type 0x0002")
        valid = False
    elif languageCode < 1 or languageCode > 3:
        print("Language code must be 1,2, or 3")
        valid = False
    elif year > 2100:
        print("Year must be below 2100")
        valid = False
    elif month < 1 or month > 12:
        print("Month must be between 1 and 12")
        valid = False
    elif day < 1 or day > 31:
        print("Day must be between 1 and 31")
        valid = False
    elif hour < 0 or hour > 23:
        print("Hour must be between 0 and 23")
        valid = False
    elif minute < 0 or minute > 59:
        print("Minute must be between 0 and 59")
        valid = False
    elif (len(packet) - 13) != length:
        print("Length field does not equal payload length")
        valid = False

    return valid



def constructReqPkt(reqType):
    """Constructs a request packet to be sent to the server. Takes a string
    reqType which is filled in the packet header, indicating date or time.
    Returns the bytearray request packet."""
    reqPkt = bytearray(6)
    magicNumber = 0x497E
    packetType = 0x0001
    bufferSize = 1024
    timeLimit = 1
```

```
    if reqType == "date":
        requestType = 0x0001
    else:
        requestType = 0x0002

    reqPkt[0] = (magicNumber >> 8) & 0xff
    reqPkt[1] = magicNumber & 0xff
    reqPkt[2] = (packetType >> 8) & 0xff
    reqPkt[3] = packetType & 0xff
    reqPkt[4] = (requestType >> 8) & 0xff
    reqPkt[5] = requestType & 0xff
    return reqPkt



def printPacket(packet):
    """Takes the server's response packet and prints the contents, formatted."""
    magicNumber = packet[0] << 8 | packet[1]
    packetType = packet[2] << 8 | packet[3]
    languageCode = packet[4] << 8 | packet[5]
    year = packet[6] << 8 | packet[7]
    month = packet[8]
    day = packet[9]
    hour = packet[10]
    minute = packet[11]
    length = packet[12]
    text = packet[13:].decode()

    print("Printing response packet contents...")
    print("Magic number = {}".format(hex(magicNumber)))
    print("Packet type = {}".format(packetType))
    print("Language code = {}".format(languageCode))
    print("Year = {}".format(year))
    print("Month = {}".format(month))
    print("Day = {}".format(day))
    print("Hour = {}".format(hour))
    print("Minute = {}".format(minute))
    print("Length = {}".format(length))
    print("Text = {}".format(text))



def receivePacket(client, timeLimit, bufferSize):
    """Takes the socket, time limit for how long to wait for a response message
    and size of the socket buffer. Either prints an error message or prints
    the packet contents."""
    readList, writeList, exception = select.select([client], [], [], timeLimit)
    if len(readList) > 0:
        responsePacket, addr = client.recvfrom(bufferSize)
        packetValid = isValidResponse(responsePacket)
        if packetValid:
            print("Response packet received")
            printPacket(responsePacket)
        else:
            print("Packet not valid")
            return
    else:
        print('Response packet was not received within {} seconds'.format(timeLimit)
)
        return



def runClient(requestType, host, port):
    """Sends a request packet to a server for either date or time. Returns the
    response packet payload if received within the timelimit, otherwise returns
    an error message."""
    timeLimit = 1
    bufferSize = 1024
    # Error checking for parameters
    if requestType != "date" and requestType != "time":
        print("Request type must be for date or time.")
```

```
        return
    elif port < 1024 or port > 64000:
        print("Port num must be between 1024 and 64000")
        return
    try:
        addrInfo = socket.getaddrinfo(host, port)
        serverAddress = addrInfo[0][4]
    except:
        print('Invalid host')
        return
    # Construct a request packet to be sent to the server
    reqPkt = constructReqPkt(requestType)
    client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    client.sendto(reqPkt, serverAddress)
    print("Request packet sent")
    # Receive the response packet from the server and prints the contents
    receivePacket(client, timeLimit, bufferSize)



def main():
    name, reqType, IP, port = argv
    runClient(reqType, IP, int(port))
main()
```