Plagiarism Declaration

This form needs to accompany your COSC 264 assignment submission.

I understand that plagiarism means taking someone else's work (text, program code, ideas, concepts) and presenting them as my own, without proper attribution. Taking someone else's work can include verbatim copying of text, figures/images, or program code, or it can refer to the extensive use of someone else's original ideas, algorithms or concepts.

I hereby declare that:

- My assignment is my own original work. I have not reproduced or modified code, figures/images, or writings of others without proper attribution. I have not used original ideas and concepts of others and presented them as my own.
- I have not allowed others to copy or modify my own code, figures/images, or writings. I have not allowed others to use original ideas and concepts of mine and present them as their own.
- I accept that plagiarism can lead to consequences, which can include partial or total loss of marks, no grade being awarded and other serious consequences, including notification of the University Proctor.

Name: Jemin Lee

Student ID: 75487642

Signature: 

Date: 21 Aug 2018.

```
import socket
import sys
import struct

class DT_request(object):
    def __init__(self, magicNo, packetType, requestType):
        self.magicNo = magicNo
        self.packetType = packetType
        self.requestType = requestType
        self.request_packet = bytearray()


    def validity_check(self):
        if self.magicNo == 0x497E:
            if self.packetType == 0x0001:
                if self.requestType == 0x0001 or self.requestType == 0x0002:
                    self.input_valid = True
                else:
                    self.input_valid = False
            else:
                self.input_valid = False
        else:
            self.input_valid = False

        return self.input_valid


    def encode(self):
        if self.input_valid == True:
            self.request_packet = struct.pack("<hhh", self.magicNo, self.packetType, self.requestType)
            result = self.request_packet
        else:
            result = "Check the input."
        return result


def main():
    try:
        input_request_type = sys.argv[1]
        UDP_ip = sys.argv[2]
        UDP_port = int(sys.argv[3])
    except IndexError:
        print("Element(s) missing")
        sys.exit()

    try:
        UDP_ip = socket.gethostbyname(UDP_ip)
    except socket.gaierror:
        print("Invalid server")
        sys.exit()

    except UnicodeError:
        print("Invalid server")
        sys.exit()


    if input_request_type == "date":
        request_type = 0x0001
    elif input_request_type == "time":
        request_type = 0x0002

    message = DT_request(0x497E, 0x0001, request_type)
    message.validity_check()
    message.encode()
    try:
        clientsock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        clientsock.sendto(message.request_packet, (UDP_ip, UDP_port))
        clientsock.settimeout(1)
    except socket.error:
        print("Error in Server")
        sys.exit()
    try:
        data, addr = clientsock.recvfrom(1024)
    except socket.timeout:
        print("Time out")
        sys.exit()
```

```
    header = data[:13]
    payload = data[13:]
    magicNo, packetType, languageCode, year, month, day, hour, minute, length = struct.unpack("<hhhhbbbbb", header)
    print("_____")
    print(data)
    print("_____")
    print(payload.decode())



main()
```

```
import socket
import sys
import datetime
import struct
import select


class DT_response(object):
    def __init__(self, request_type, languageCode):
        #date and time
        self.now = datetime.datetime.now()
        self.year = self.now.year
        self.month = self.now.month
        self.day = self.now.day
        self.hour = self.now.hour
        self.minute = self.now.minute
        self.english = {1: "January", 2: "February", 3: "March", 4: "April", 5: "May", 6: "June", 7: "July", 8: "August", 9: "September", 10: "October", 11:
"November", 12: "December"}
        self.maori = {1: "Kohit¯atea", 2: "Hui-tanguru", 3: "Pout¯u-te-rangi", 4: "Paenga-wh¯awh¯a", 5: "Haratua", 6: "Pipiri", 7: "H¯ongongoi", 8: "Here-turi-
k¯ok¯a", 9: "Mahuru", 10: "Whiringa-¯a-nuku", 11: "Whiringa-¯a-rangi", 12: "Hakihea"}
        self.german = {1: "Januar", 2: "Februar", 3: "M¨arz", 4: "April", 5: "Mai", 6: "Juni", 7: "Juli", 8: "August", 9: "September", 10: "Oktober", 11:
"November", 12: "Dezember"}

        #packet components
        self.magicNo = 0x497E
        self.packetType = 0x0002
        self.request_type = request_type
        self.languageCode = languageCode
        self.response_packet = bytearray()


    def payload_string(self):
        if self.request_type == 0x0001: #date
            if self.languageCode == 0x0001:
                self.payload = "Today's date is %s %d, %d" % (self.english[self.month], self.day, self.year)
                print(self.payload)#test
            elif self.languageCode == 0x0002:
                self.payload = "Ko te ra o tenei ra ko %s %d, %d" % (self.maori[self.month], self.day, self.year)
                print(self.payload)#test
            elif self.languageCode == 0x0003:
                self.payload = "Heute ist der %s %d, %d" % (self.german[self.month], self.day, self.year)
                print(self.payload)#test
        elif self.request_type == 0x0002: #time
            if self.languageCode == 0x0001:
                self.payload = "The current time is %d:%d" % (self.hour, self.minute)
                print(self.payload)#test
            elif self.languageCode == 0x0002:
                self.payload = "Ko te wa o tenei wa %d:%d" % (self.hour, self.minute)
                print(self.payload)#test
            elif self.languageCode == 0x0003:
                self.payload = "Die Uhrzeit ist %d:%d" % (self.hour, self.minute)
                print(self.payload)#test
        return self.payload

    def payload_length_check(self):
        self.length = len(self.payload.encode())
        if self.length >= 255:
            print("_____")
            print("ERROR: length too long")
            return False
        print("valid length")
        print("_____")
        return True

    def packet_encode(self):
        self.payload_string()
        self.payload_byte = self.payload.encode('utf-8')
        if self.payload_length_check():
            self.response_packet_header = struct.pack(">hhhhbbbbb",self.magicNo, self.packetType, self.languageCode, self.year, self.month, self.day,
self.hour, self.minute, self.length)
            self.response_packet = self.response_packet_header + self.payload_byte
            result = self.response_packet
            print(self.response_packet)
        return result
```

```
def request_packet_check(data):
    magicNo, packetType, request_type = struct.unpack("<hhh", data)
    packet_length = struct.calcsize(">hhh")
    if packet_length == 6:
        if (magicNo == 0x497E) and (packetType == 0x0001) and (request_type == 0x0001 or request_type == 0x0002):
            validity_check = True
    else:
        validity_check = False
    return validity_check

def decode(data):
    if request_packet_check(data):
        magicNo, packetType, request_type = struct.unpack("<hhh", data)
    return request_type

def main():
    UDP_ip = "127.0.0.1"
    try:
        UDP_port_eng = int(sys.argv[1])
        UDP_port_mao = int(sys.argv[2])
        UDP_port_ger = int(sys.argv[3])
    except IndexError:
        print("_____")
        print("Element(s) missing")
        sys.exit()

    try:
        #english packet
        sock_eng = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        sock_eng.bind((UDP_ip, UDP_port_eng))

        #maori
        sock_mao = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        sock_mao.bind((UDP_ip, UDP_port_mao))

        #german
        sock_ger = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        sock_ger.bind((UDP_ip, UDP_port_ger))
    except socket.error:
        print("_____")
        print("Port number Error")
        sys.exit()



    selecting = True
    while selecting:
        client, _, _ = select.select([sock_eng, sock_mao, sock_ger], [], [], None)
        for s in client:
            if s:
                data, addr = s.recvfrom(1024)
                if s == sock_eng:
                    print("_____")
                    print ("English", data)
                    print("Packet length : " , len(data))
                    languageType = 0x0001

                elif s == sock_mao:
                    print("_____")
                    print ("Maori", data)
                    print("Packet length : " , len(data))
                    languageType = 0x0002

                elif s == sock_ger:
                    print("_____")
                    print ("German", data)
                    print("Packet length : " , len(data))
                    languageType = 0x0003
        request_port_num = s.getsockname()[1]
        #setting and passing request_type and language_type to DT_response class
        requestType = decode(data)
        p = DT_response(requestType, languageType)
        p.packet_encode()
        server_response = p.response_packet
```

```
        selecting = False
        while True:
            s.sendto(server_response, (UDP_ip, addr[1]))
            print("_____")
            print("sent, terminate")
            break
main()
```