
LABORATORY-3
SOCKET PROGRAMMING

Q1) Auction System with Multiple Clients
#server.py

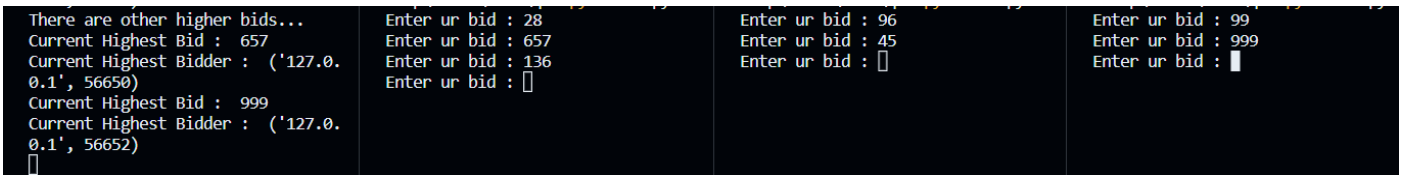
```
import socket, time, threading
HOST = "127.0.0.1"
PORT = 12345
print("Auction Starts......")
base = int(input("Enter Base Rate : "))
n = int(input("Number of Participants : "))

current_highest_bid = base
current_highest_bidder = None
bidLock = threading.Lock()

def set_idle_timeout(sock, timeout, addr):
    global current_highest_bid
    global current_highest_bidder

    sock.settimeout(timeout)
    last_activity = time.time()

    while True:
        try:
            data = int(sock.recv(1024).decode())
            last_activity = time.time()
            with bidLock:
                if(data > current_highest_bid):
                    current_highest_bid = data
                    current_highest_bidder = addr

                else:
                    print("There are other higher bids...")
            print("Current Highest Bid : ", current_highest_bid)
            print("Current Highest Bidder : ", current_highest_bidder)

        except socket.timeout or socket.error:
            if time.time() - last_activity > timeout:
                break

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((HOST, PORT))
s.listen(n+1)

l=[]
for i in range(n):
    conn, addr = s.accept()
```

```
    t = threading.Thread(target=set_idle_timeout, args=(conn, 30, addr))
    t.start()
    l.append(t)

for thread in l:
    thread.join()

print("Auction Over...")
print("Highest Bid : ", current_highest_bid)
```

#client.py

```
import socket
HOST = "127.0.0.1"
PORT = 12345
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    while True:
        try:
            bid = input("Enter ur bid : ")
            s.send(bid.encode())
        except:
            print("Auction Over")
            break
```

I/O :



Q2) Hexapawn game

#server.py

```
import socket
HOST = '127.0.0.1'
PORT = 12345
board = [['X', 'X', 'X'], [' ', ' ', ' '], ['O', 'O', 'O']]
def check(sr,sc,er,ec,board):
    if(sr < 0 or sr > 2 or sc < 0 or sc > 2 or er < 0 or er > 2 or ec < 0 or
ec > 2):
        return 0
    if(board[sr][sc] == 'X'):
        if(board[er][ec] == ' ' and (er == sr+1 and ec == sc)):
            board[sr][sc] = ' '
            board[er][ec] = 'X'
            return 1
        elif(board[er][ec] == 'O' and ((er == sr+1 and ec == sc+1) or (er ==
sr+1 and ec == sc-1))):
            board[sr][sc] = ' '
            board[er][ec] = 'X'
            return 1
        else:
            return 0
    elif(board[sr][sc] == 'O'):
        if(board[er][ec] == ' ' and (er == sr-1 and ec == sc)):
            board[sr][sc] = ' '
            board[er][ec] = 'O'
            return 1
        elif(board[er][ec] == 'X' and ((er == sr-1 and ec == sc+1) or (er ==
sr-1 and ec == sc-1))):
            board[sr][sc] = ' '
            board[er][ec] = 'O'
            return 1
        else:
            return 0
    else:
        return 0

def checkstatus(board):
    for i in range(3):
        if(board[0][i] == 'O'):
            return 1
        elif(board[2][i] == 'X'):
            return 2
    return 0

def checkdraw(board):
    for i in range(3):
```

```
        for j in range(3):
            if(board[i][j] == 'X'):
                if((i+1 < 3 and board[i+1][j] == 'O') or (i+1 < 3 and j+1 < 3
and board[i+1][j+1] == ' ') or(i+1 < 3 and j-1 >= 0 and board[i+1][j-1] == '
')):
                    continue
                else:
                    return 0
            elif(board[i][j] == 'O'):
                if((i-1 >= 0 and board[i-1][j] == 'X') or (i-1 >= 0 and j+1 <
3 and board[i-1][j+1] == ' ') or(i-1 >= 0 and j-1 >= 0 and board[i-1][j-1] ==
' ')):
                    continue
                else:
                    return 0
    return 1
print(board)
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    conn1, addr1 = s.accept()
    conn2, addr2 = s.accept()
    while True:
        m1 = conn1.recv(1024).decode()
        sr,sc,er,ec = map(int, m1.split())
        flag = check(sr,sc,er,ec,board)
        if(flag == 0):
            conn1.send("100".encode())
            continue
        print(board)
        m2 = conn2.recv(1024).decode()
        sr,sc,er,ec = map(int, m2.split())
        flag = check(sr,sc,er,ec,board)
        if(flag == 0):
            conn2.send("100".encode())
            continue
        print(board)
        status = checkstatus(board)
        if(status == 1):
            conn1.send("300".encode())
            break
        elif(status == 2):
            conn2.send("300".encode())
            break
        draw = checkdraw(board)
        if(draw == 1):
            conn1.send("400".encode())
            conn2.send("400".encode())
```

```
            break
        conn1.send("200".encode())
        conn2.send("200".encode())
```

#client.py

```
import socket

HOST = '127.0.0.1'
PORT = 12345

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    while True:
        data = input("Enter the move : ")
        s.send(data.encode())
        res = s.recv(1024).decode()
        if(res == "100"):
            print("Invalid Move")
        elif(res == "300"):
            print("You Won")
            break
        elif(res == "400"):
            print("Draw")
            break
        else:
            continue
```

I/O: