CHECK SUM IN C++

```
//Check Sum
#include <bits/stdc++.h>
using namespace std;
string Ones_complement(string data)
{
for (int i = 0; i < data.length(); i++) {
      if (data[i] == '0')
            data[i] = '1';
      else
            data[i] = '0';
}
return data;
}
string checkSum(string data, int block_size)
{
int n = data.length();
if (n % block_size != 0) {
      int pad_size = block_size - (n % block_size);
      for (int i = 0; i < pad_size; i++)
            data = '0' + data;
}
string result = "";
for (int i = 0; i < block_size; i++)
      result += data[i];
for (int i = block_size; i < n; i += block_size) {
      string next_block = "";
      for (int j = i; j < i + block_size; j++)
            next_block += data[j];
      string additions = "";
      int sum = 0, carry = 0;
      for (int k = block_size - 1; k >= 0; k--) {
            sum += (next_block[k] - '0')+ (result[k] -
'0');
            carry = sum / 2;
            if (sum == 0)
                  {     additions = '0' + additions; sum =
carry;      }
            else if (sum == 1)
                  {     additions = '1' + additions; sum =
carry;      }
            else if (sum == 2)
                  {     additions = '0' + additions; sum =
carry;      }
```

```
          else {     additions = '1' + additions; sum =
carry;     }
     }
     string final = "";
     if (carry == 1) {
          for (int l = additions.length() - 1; l >=
0;l--) {
               if (carry == 0)
                    final = additions[l] + final;
               else if (((additions[l] - '0') + carry) %
2== 0)
               {    final = "0" + final;carry = 1;   }
               else
               {    final = "1" + final;carry = 0;   }
          }
          result = final;
     }
     else {  result = additions;    }
}
return Ones_complement(result);
}
bool checker(string sent_message,string rec_message,int
block_size){
string sender_checksum= checkSum(sent_message,
block_size);
string receiver_checksum = checkSum(rec_message +
sender_checksum, block_size);
if
(count(receiver_checksum.begin(),receiver_checksum.end(),
'0')== block_size)
    { return true;    }
else {  return false; }
}
int main()
{
string sent_message= "10000101011000111001010011101101";
string recv_message= "10000101011000111001010011101101";
int block_size = 8;
if (checker(sent_message,recv_message,block_size))
    {      cout << "No Error";   }
else {     cout << "Error"; }
return 0;
}
```

TIME SERVER IN C++

```
//Time Server
#include <iostream>
#include <ctime>
#include <cstring>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <unistd.h>
int main()
{
    //Create a TCP socket
    int serverSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (serverSocket == -1)
    {
        std::cerr << "Failed to create socket." <<
std::endl;
        return 1;
    }

    //Bind the socket to a port
    sockaddr_in serverAddress{};
    serverAddress.sin_family = AF_INET;
    serverAddress.sin_addr.s_addr = INADDR_ANY;
    serverAddress.sin_port = htons(8888);
    if (bind(serverSocket, reinterpret_cast<struct
sockaddr *>(&serverAddress), sizeof(serverAddress)) < 0)
   {
        std::cerr << "Failed to bind socket." <<
std::endl;
        return 1;
    }

    //Listen for incoming connections
    listen(serverSocket, 1);
    std::cout << "Server started. Listening on port
8888..." << std::endl;
    while (true)
   {
        //Accept a client connection
        int clientSocket = accept(serverSocket, nullptr,
nullptr);
```

```
        if (clientSocket < 0)
       {
            std::cerr << "Failed to accept client
connection." << std::endl;
            continue;
        }

        //Get the current time
        std::time_t currentTime = std::time(nullptr);
        std::string timeString =
std::ctime(&currentTime);

        //Send the time to the client
        if (send(clientSocket, timeString.c_str(),
timeString.length(), 0) < 0)
       {
            std::cerr << "Failed to send time to
client." << std::endl;
        }
        close(clientSocket);
    }
    close(serverSocket);
    return 0;
}
```

TIME CLIENT IN C++

```
#include <iostream>
#include <cstring>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <unistd.h>

int main()
{
    // Create a TCP socket
    int clientSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (clientSocket == -1)
   {
        std::cerr << "Failed to create socket." <<
std::endl;
        return 1;
    }

    // Specify the server address and port
    sockaddr_in serverAddress{};
    serverAddress.sin_family = AF_INET;
    serverAddress.sin_port = htons(8888);
    if (inet_pton(AF_INET, "127.0.0.1",
&(serverAddress.sin_addr)) <= 0) {
        std::cerr << "Invalid address/Address not
supported." << std::endl;
        return 1;
    }

    // Connect to the server
    if (connect(clientSocket, reinterpret_cast<struct
sockaddr *>(&serverAddress), sizeof(serverAddress)) < 0)
   {
        std::cerr << "Connection failed." << std::endl;
        return 1;
    }

    // Receive the time from the server
    char buffer[1024];
    memset(buffer, 0, sizeof(buffer));
    if (recv(clientSocket, buffer, sizeof(buffer), 0) < 0)
   {
        std::cerr << "Failed to receive time from server."
<< std::endl;
        return 1;
    }

    // Print the received time
    std::cout << "Current time: " << buffer;
    close(clientSocket);
    return 0;
}
```

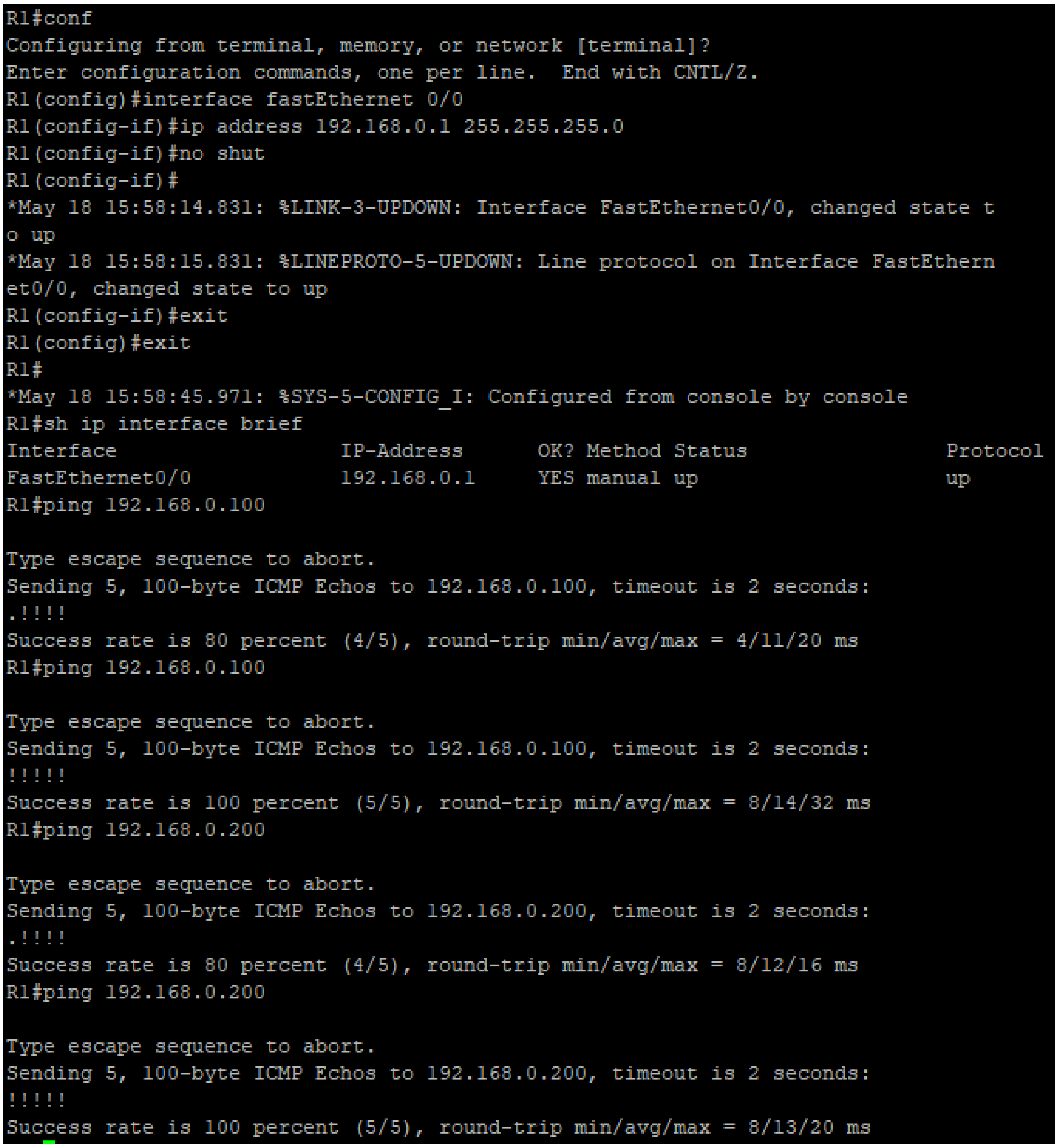
MATH SERVER IN C++

```
#include <iostream>
#include <cstdlib>
#include <cstring>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define PORT 12345
int performOperation(int num1, int num2, char op) {
switch (op) {
case '+':  return num1 + num2;
case '-':  return num1 - num2;
case '*': return num1 * num2;
case '/':
if (num2 != 0)
return num1 / num2;
else
return 0;
default: eturn 0;
}
}
int main() {
int serverSocket, clientSocket;char buffer[1024];
struct sockaddr_in serverAddr, clientAddr;

// Create socket
serverSocket = socket(AF_INET, SOCK_STREAM, 0);
if (serverSocket == -1)
{std::cerr << "Error creating socket.\n";return 1;}

// Prepare the sockaddr_in structure
serverAddr.sin_family = AF_INET;
serverAddr.sin_addr.s_addr = INADDR_ANY;
serverAddr.sin_port = htons(PORT);

//Bind
if (bind(serverSocket, (struct sockaddr *) &serverAddr,
               sizeof(serverAddr)) < 0)
    {std::cerr << "Error binding.\n";return 1;     }
// Listen

if (listen(serverSocket, 3) < 0)
    {std::cerr << "Error listening.\n";return 1;   }
std::cout << "Math server is running. Waiting for client
               connections...\n";
socklen_t addrLen = sizeof(clientAddr);

// Accept connection from incoming client
clientSocket = accept(serverSocket, (struct sockaddr *)
               &clientAddr, &addrLen);
if (clientSocket < 0)
{std::cerr << "Error accepting connection.\n";return 1;  }

// Read data from client
memset(buffer, 0, sizeof(buffer));
if (read(clientSocket, buffer, sizeof(buffer)) < 0)
{std::cerr << "Error reading from socket.\n";return 1; }
int num1, num2, result;char op;

// Parse data received from client
sscanf(buffer, "%d %c %d", &num1, &op, &num2);
result = performOperation(num1, num2, op);
std::cout << "Performing operation: " << num1 << " " << op << "
               " << num2 << "\n";
std::cout << "Result: " << result << "\n";

// Convert result to string
std::string resultStr = std::to_string(result);

// Send result back to client
if (send(clientSocket, resultStr.c_str(), resultStr.size(), 0) <
               0)
{ std::cerr << "Error sending data.\n";return 1;}

// Close sockets
close(clientSocket);
close(serverSocket);
return 0;
}
```

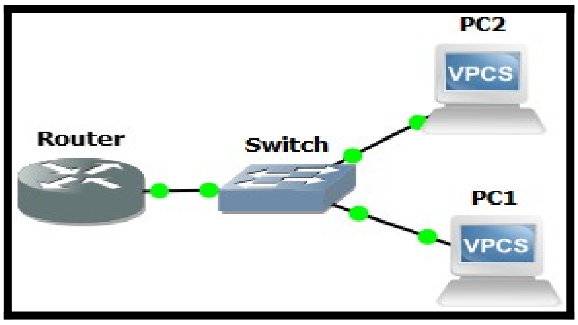
MATH CLIENT IN C++

```
#include <iostream>
#include <cstdlib>
#include <cstring>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define PORT 12345
int main() {
 int clientSocket;char buffer[1024];
 struct sockaddr_in serverAddr;

 // Create socket
 clientSocket = socket(AF_INET, SOCK_STREAM, 0);
 if (clientSocket == -1)
 { std::cerr << "Error creating socket.\n";return 1;}

// Prepare the sockaddr_in structure
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons(PORT);

// Convert IP address from string to binary form
if (inet_pton(AF_INET, "127.0.0.1", &(serverAddr.sin_addr)) <=
0)
{std::cerr << "Invalid address/Address not supported.\n";return
1;}

// Connect to server
if (connect(clientSocket, (struct sockaddr *) &serverAddr,
sizeof(serverAddr)) < 0)
{std::cerr << "Connection failed.\n";return 1;}
int num1, num2;
char op;
std::cout << "Enter first number: ";
std::cin >> num1;
std::cout << "Enter operator (+, -, *, /): ";
std::cin >> op;
std::cout << "Enter second number: ";
std::cin >> num2;
// Prepare message to send to server
std::string message = std::to_string(num1) + " " + op + " " +
std::to_string(num2);

// Send data to server
if (send(clientSocket, message.c_str(), message.size(), 0) < 0)
{std::cerr << "Error sending data.\n";return 1;}

// Receive result from server
memset(buffer, 0, sizeof(buffer));
if (read(clientSocket, buffer, sizeof(buffer)) < 0)
{std::cerr << "Error receiving data.\n";return 1; }
std::cout << "Result received from server: " << buffer << "\n";

// Close socket
close(clientSocket);return 0;
}
```

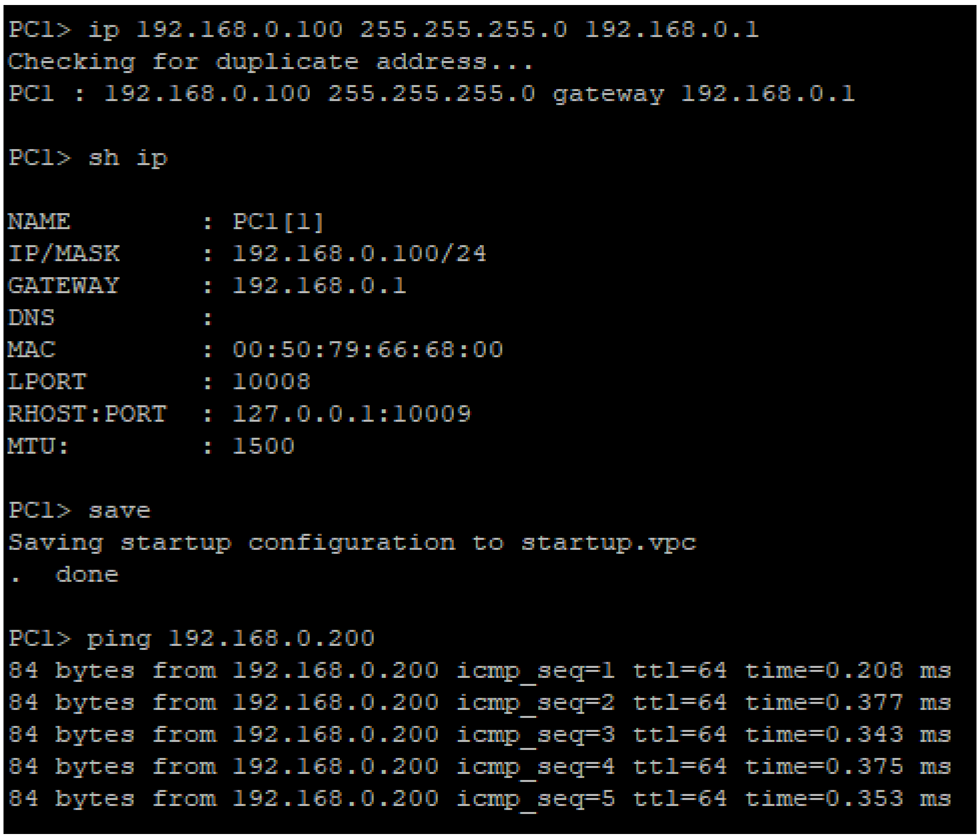
GNS3 Router COMMANDS



```
R1#conf
Configuring from terminal, memory, or network [terminal]?
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface fastEthernet 0/0
R1(config-if)#ip address 192.168.0.1 255.255.255.0
R1(config-if)#no shut
R1(config-if)#
*May 18 15:58:14.831: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state t
o up
*May 18 15:58:15.831: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthern
et0/0, changed state to up
R1(config-if)#exit
R1(config)#exit
R1#
*May 18 15:58:45.971: %SYS-5-CONFIG_I: Configured from console by console
R1#sh ip interface brief
Interface                  IP-Address      OK? Method Status                Protocol
FastEthernet0/0            192.168.0.1     YES manual up                    up
R1#ping 192.168.0.100

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.0.100, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 4/11/20 ms
R1#ping 192.168.0.100

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.0.100, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/14/32 ms
R1#ping 192.168.0.200

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.0.200, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 8/12/16 ms
R1#ping 192.168.0.200

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.0.200, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/13/20 ms
```

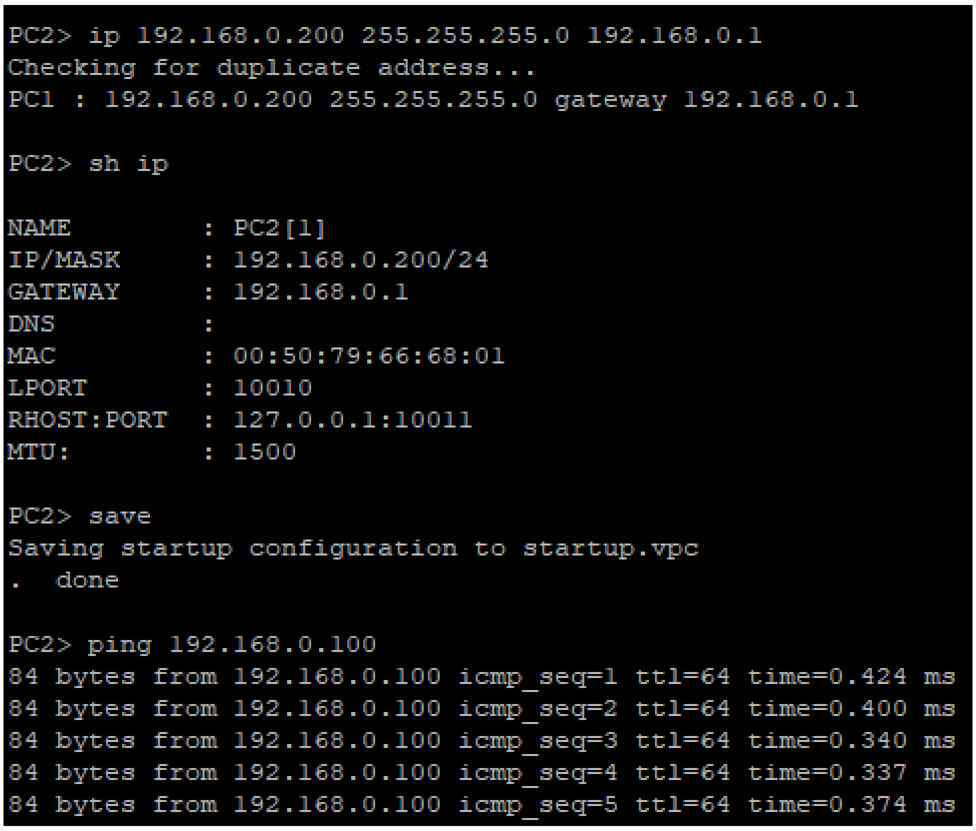


GNS3 PC1 COMMANDS



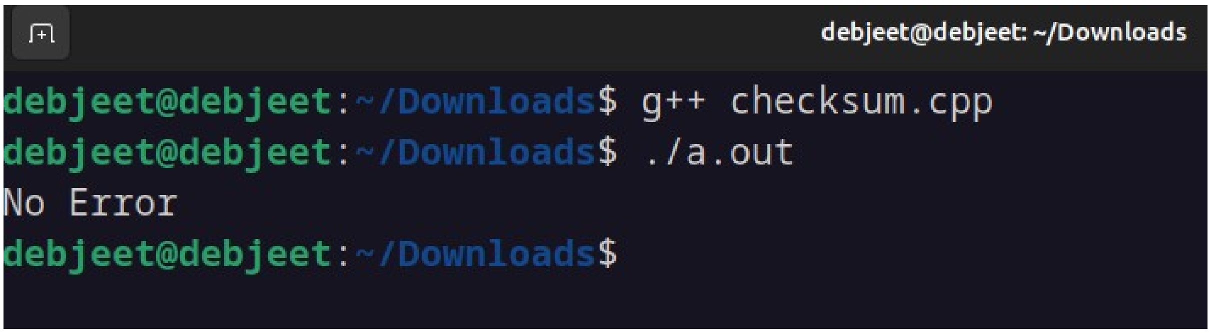
```
PC1> ip 192.168.0.100 255.255.255.0 192.168.0.1
Checking for duplicate address...
PC1 : 192.168.0.100 255.255.255.0 gateway 192.168.0.1

PC1> sh ip

NAME         : PC1[1]
IP/MASK      : 192.168.0.100/24
GATEWAY      : 192.168.0.1
DNS          :
MAC          : 00:50:79:66:68:00
LPORT        : 10008
RHOST:PORT   : 127.0.0.1:10009
MTU:         : 1500

PC1> save
Saving startup configuration to startup.vpc
.  done

PC1> ping 192.168.0.200
84 bytes from 192.168.0.200 icmp_seq=1 ttl=64 time=0.208 ms
84 bytes from 192.168.0.200 icmp_seq=2 ttl=64 time=0.377 ms
84 bytes from 192.168.0.200 icmp_seq=3 ttl=64 time=0.343 ms
84 bytes from 192.168.0.200 icmp_seq=4 ttl=64 time=0.375 ms
84 bytes from 192.168.0.200 icmp_seq=5 ttl=64 time=0.353 ms
```

GNS3 PC2 COMMANDS



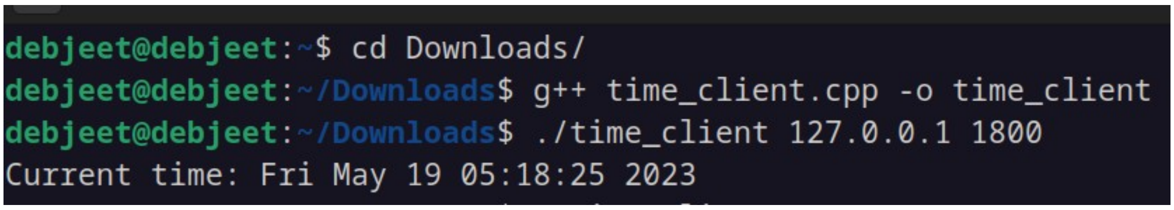
```
PC2> ip 192.168.0.200 255.255.255.0 192.168.0.1
Checking for duplicate address...
PC1 : 192.168.0.200 255.255.255.0 gateway 192.168.0.1

PC2> sh ip

NAME         : PC2[1]
IP/MASK      : 192.168.0.200/24
GATEWAY      : 192.168.0.1
DNS          :
MAC          : 00:50:79:66:68:01
LPORT        : 10010
RHOST:PORT   : 127.0.0.1:10011
MTU:         : 1500

PC2> save
Saving startup configuration to startup.vpc
.  done

PC2> ping 192.168.0.100
84 bytes from 192.168.0.100 icmp_seq=1 ttl=64 time=0.424 ms
84 bytes from 192.168.0.100 icmp_seq=2 ttl=64 time=0.400 ms
84 bytes from 192.168.0.100 icmp_seq=3 ttl=64 time=0.340 ms
84 bytes from 192.168.0.100 icmp_seq=4 ttl=64 time=0.337 ms
84 bytes from 192.168.0.100 icmp_seq=5 ttl=64 time=0.374 ms
```

CHECKSUM-

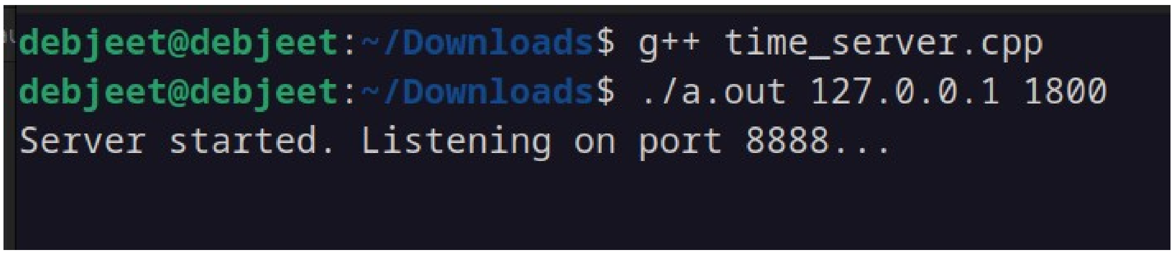


```
debjeet@debjeet:~/Downloads$ g++ checksum.cpp
debjeet@debjeet:~/Downloads$ ./a.out
No Error
debjeet@debjeet:~/Downloads$
```

TIME S&C-

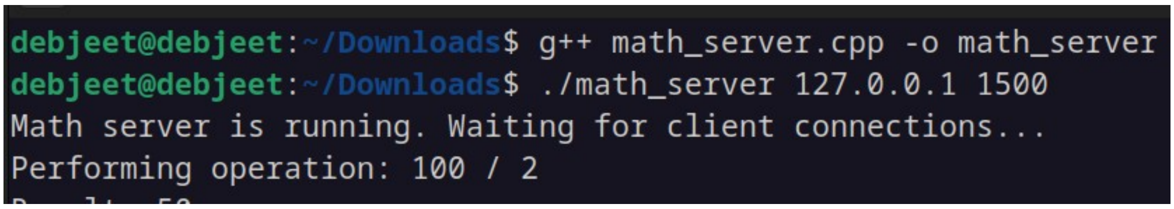


```
debjeet@debjeet:~$ cd Downloads/
debjeet@debjeet:~/Downloads$ g++ time_client.cpp -o time_client
debjeet@debjeet:~/Downloads$ ./time_client 127.0.0.1 1800
Current time: Fri May 19 05:18:25 2023
```

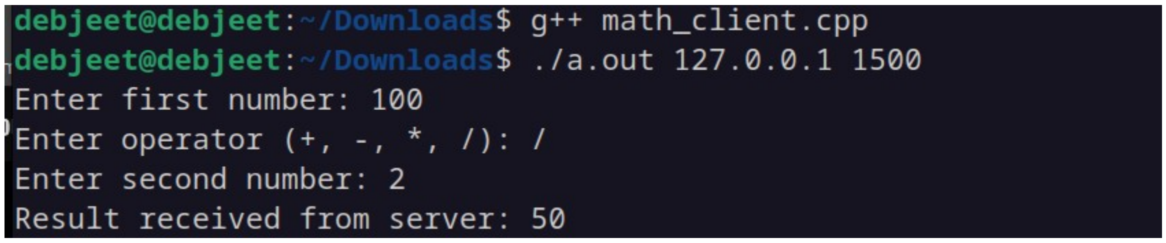


```
debjeet@debjeet:~/Downloads$ g++ time_server.cpp
debjeet@debjeet:~/Downloads$ ./a.out 127.0.0.1 1800
Server started. Listening on port 8888...
```

MATH S&C-



```
debjeet@debjeet:~/Downloads$ g++ math_server.cpp -o math_server
debjeet@debjeet:~/Downloads$ ./math_server 127.0.0.1 1500
Math server is running. Waiting for client connections...
Performing operation: 100 / 2
```



```
debjeet@debjeet:~/Downloads$ g++ math_client.cpp
debjeet@debjeet:~/Downloads$ ./a.out 127.0.0.1 1500
Enter first number: 100
Enter operator (+, -, *, /): /
Enter second number: 2
Result received from server: 50
```