Computer Networks

Name : M. Sai Saranya

Regno: 22BAI1471

Course Title : Computer Networks

Course code : BCSE308P

Slot : L45-46

Faculty : Dr Neelanarayanan V

S.No	Experiment Name	Date	Page No.	Marks
3.	**IP Address Validation and Simple application of ATM using TCP**	**24-01-2024**		

Experiment No. 3

Experiment Name: IP address validation and simulation of ATM using TCP socket-client server

Date: 24-1-2024

Problem Statement

1) Write a program to validate IP address
2) Implement a simulation of ATM functions using a TCP socket client server program

Aim

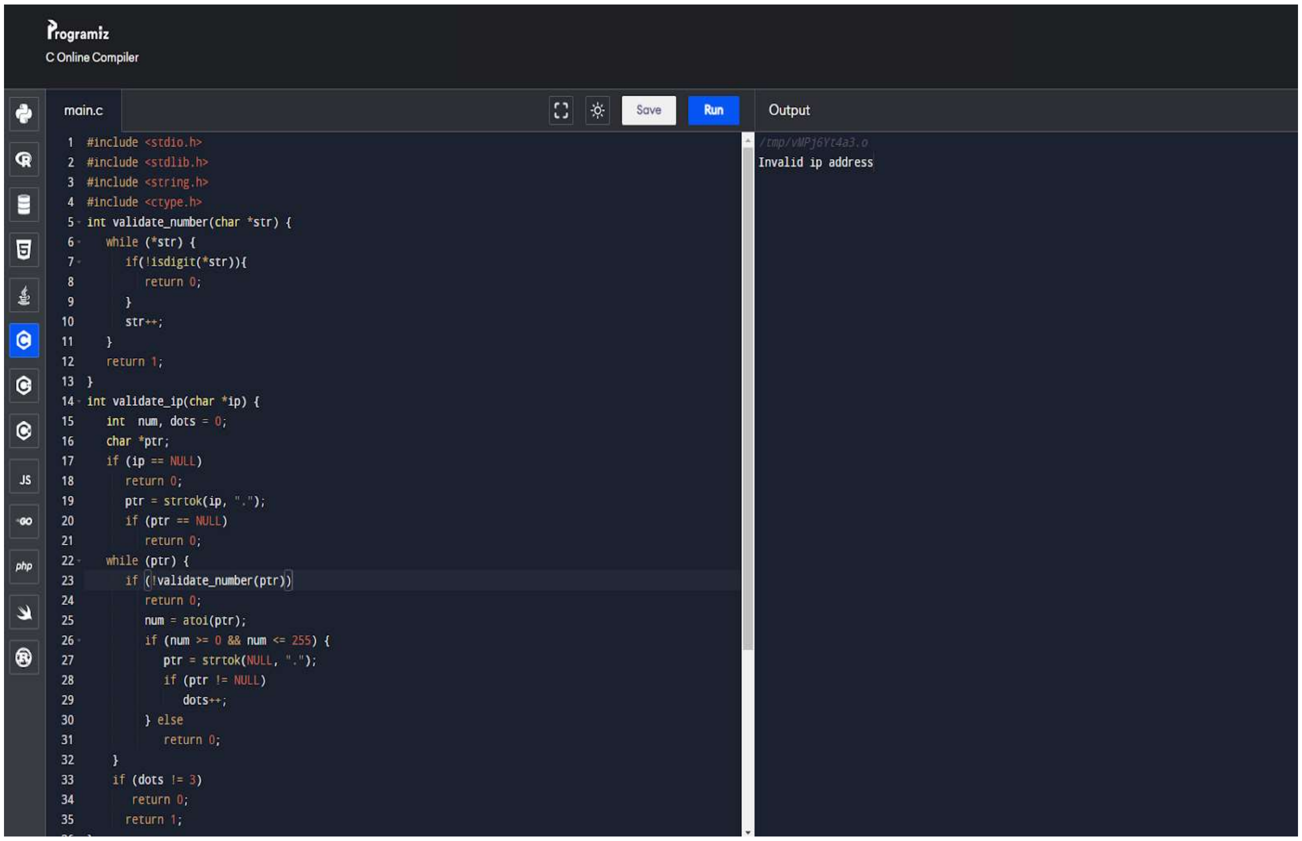
To write a c program for IP address validation and implementation of ATM basic functions using TCP socket client server program

Algorithm or Procedure

IPv4 Validation :

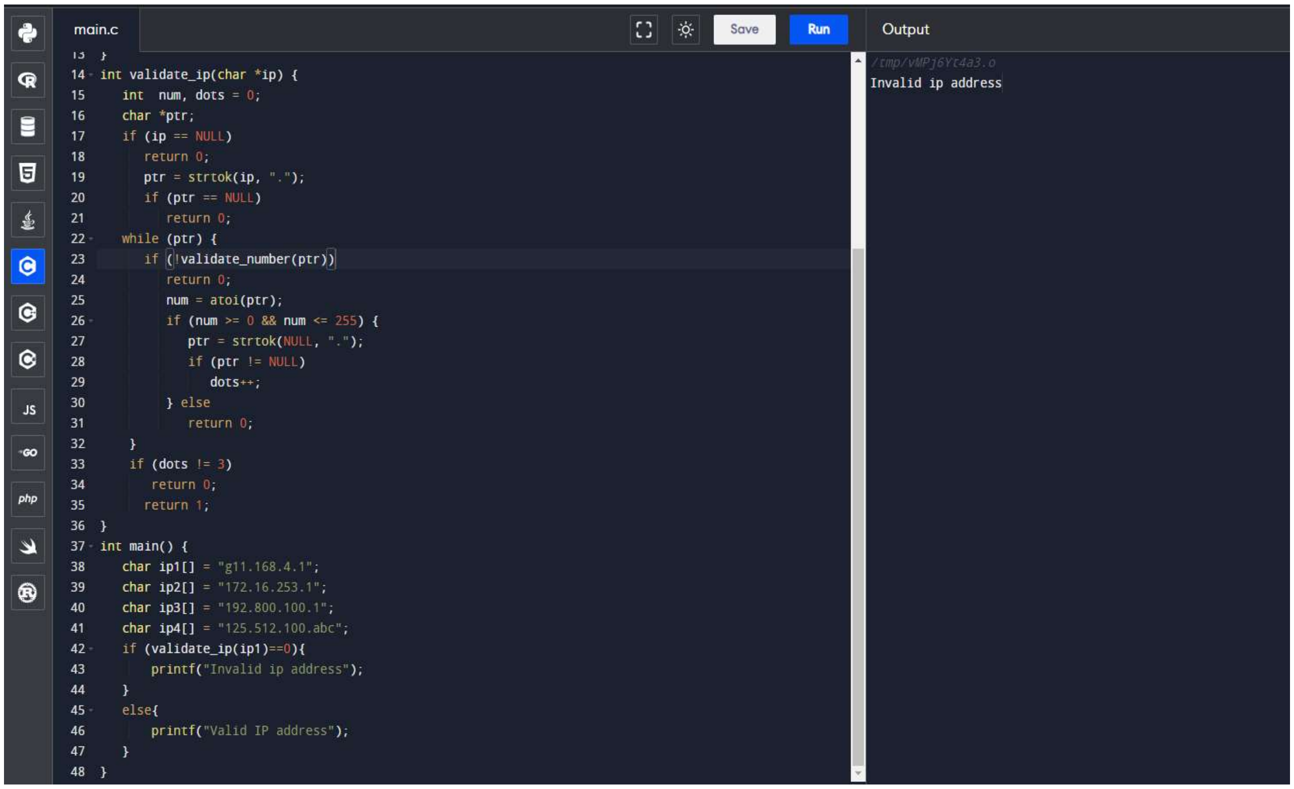
1. Split string by ., ensure exactly 4 parts.
2. Each part: convert to int, check 0-255 range.
3. No part can have leading zeros (except "0" itself).
4. No alpha characters allowed in any part.
5. If all checks pass, valid; else, invalid.

IP address validation



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
int validate_number(char *str) {
   while (*str) {
      if(!isdigit(*str)){
         return 0;
      }
      str++;
   }
   return 1;
}
int validate_ip(char *ip) {
   int  num, dots = 0;
   char *ptr;
   if (ip == NULL)
      return 0;
      ptr = strtok(ip, ".");
      if (ptr == NULL)
         return 0;
   while (ptr) {
      if (!validate_number(ptr))
         return 0;
         num = atoi(ptr);
         if (num >= 0 && num <= 255) {
            ptr = strtok(NULL, ".");
            if (ptr != NULL)
               dots++;
         } else
            return 0;
    }
    if (dots != 3)
       return 0;
      return 1;
```

```
/tmp/vMPj6Yt4a3.o
Invalid ip address
```



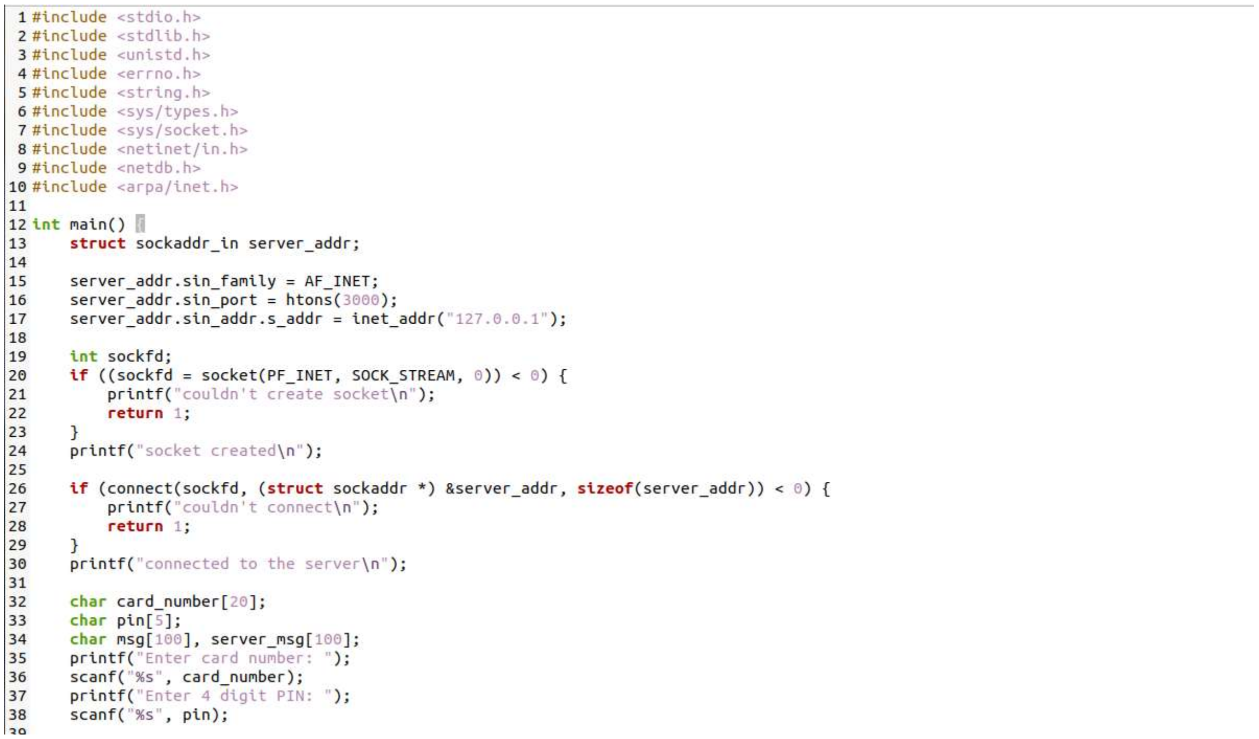
```c
}
int validate_ip(char *ip) {
   int  num, dots = 0;
   char *ptr;
   if (ip == NULL)
      return 0;
      ptr = strtok(ip, ".");
      if (ptr == NULL)
         return 0;
   while (ptr) {
      if (!validate_number(ptr))
         return 0;
         num = atoi(ptr);
         if (num >= 0 && num <= 255) {
            ptr = strtok(NULL, ".");
            if (ptr != NULL)
               dots++;
         } else
            return 0;
    }
    if (dots != 3)
       return 0;
      return 1;
```

```c
}
int main() {
   char ip1[] = "g11.168.4.1";
   char ip2[] = "172.16.253.1";
   char ip3[] = "192.800.100.1";
   char ip4[] = "125.512.100.abc";
   if (validate_ip(ip1)==0){
       printf("Invalid ip address");
   }
   else{
       printf("Valid IP address");
   }
}
```

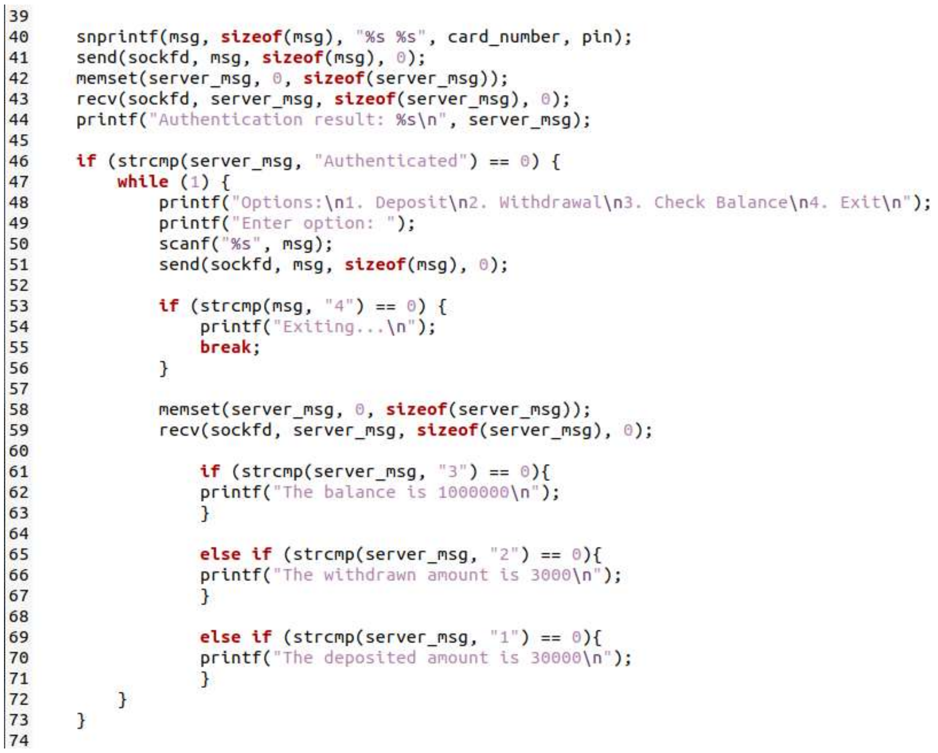
```
/tmp/vMPj6Yt4a3.o
Invalid ip address
```

ATM simulation using TCP socket client server program

Client program

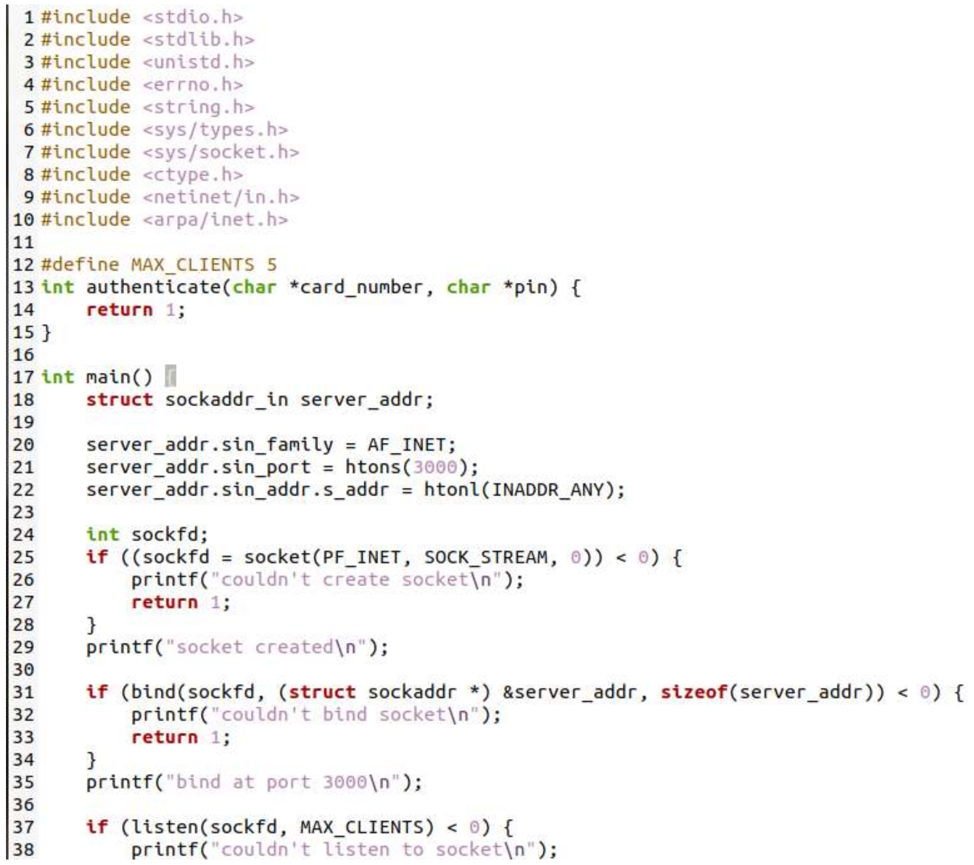


```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>

int main() {
    struct sockaddr_in server_addr;

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(3000);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    int sockfd;
    if ((sockfd = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
        printf("couldn't create socket\n");
        return 1;
    }
    printf("socket created\n");

    if (connect(sockfd, (struct sockaddr *) &server_addr, sizeof(server_addr)) < 0) {
        printf("couldn't connect\n");
        return 1;
    }
    printf("connected to the server\n");

    char card_number[20];
    char pin[5];
    char msg[100], server_msg[100];
    printf("Enter card number: ");
    scanf("%s", card_number);
    printf("Enter 4 digit PIN: ");
    scanf("%s", pin);
```

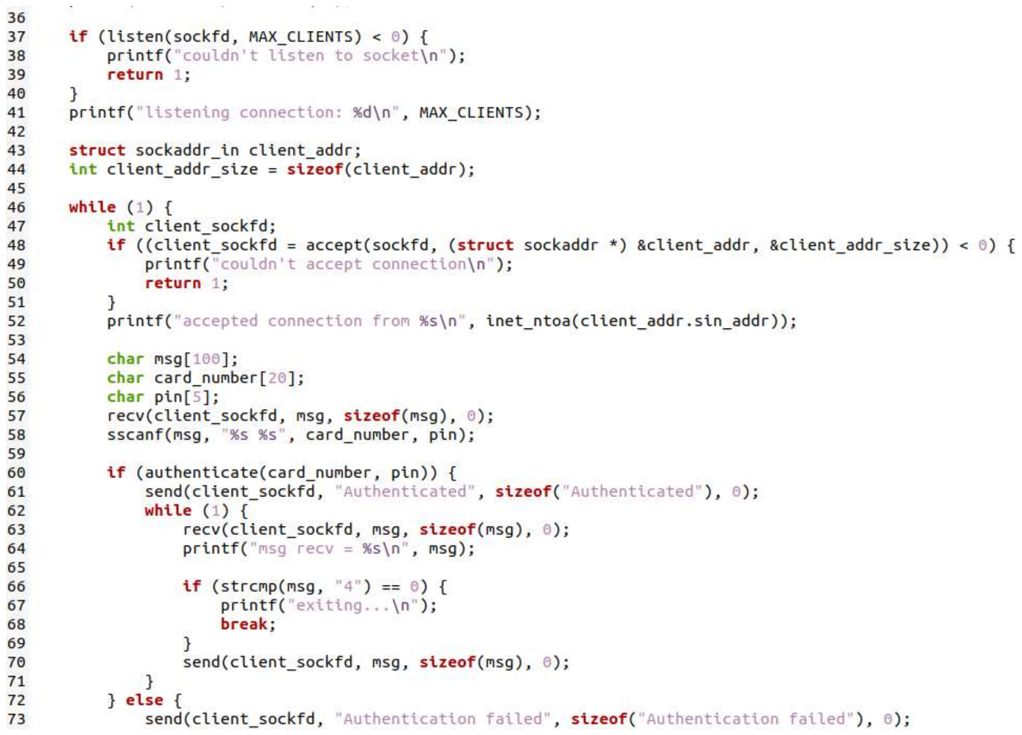


```c

    snprintf(msg, sizeof(msg), "%s %s", card_number, pin);
    send(sockfd, msg, sizeof(msg), 0);
    memset(server_msg, 0, sizeof(server_msg));
    recv(sockfd, server_msg, sizeof(server_msg), 0);
    printf("Authentication result: %s\n", server_msg);

    if (strcmp(server_msg, "Authenticated") == 0) {
        while (1) {
            printf("Options:\n1. Deposit\n2. Withdrawal\n3. Check Balance\n4. Exit\n");
            printf("Enter option: ");
            scanf("%s", msg);
            send(sockfd, msg, sizeof(msg), 0);

            if (strcmp(msg, "4") == 0) {
                printf("Exiting...\n");
                break;
            }

            memset(server_msg, 0, sizeof(server_msg));
            recv(sockfd, server_msg, sizeof(server_msg), 0);

                if (strcmp(server_msg, "3") == 0){
                printf("The balance is 1000000\n");
                }

                else if (strcmp(server_msg, "2") == 0){
                printf("The withdrawn amount is 3000\n");
                }

                else if (strcmp(server_msg, "1") == 0){
                printf("The deposited amount is 30000\n");
                }
        }
    }

```

Server program



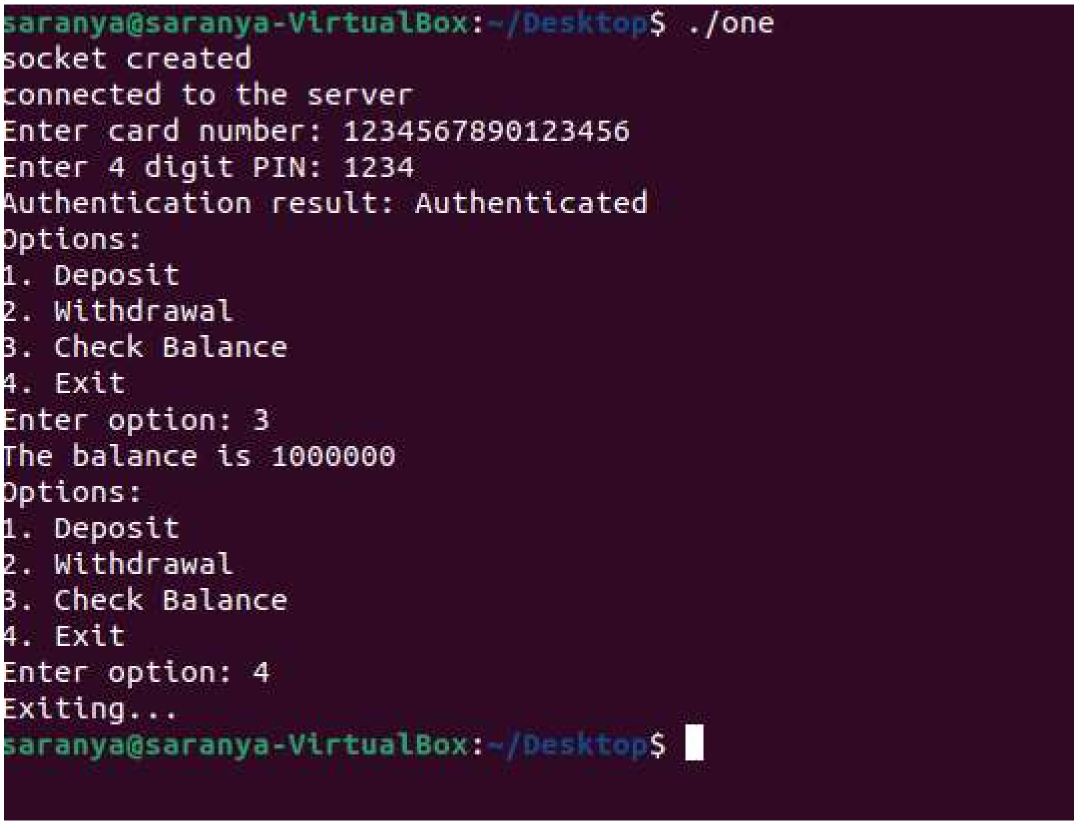
```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <ctype.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define MAX_CLIENTS 5
int authenticate(char *card_number, char *pin) {
    return 1;
}

int main() {
    struct sockaddr_in server_addr;

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(3000);
    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);

    int sockfd;
    if ((sockfd = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
        printf("couldn't create socket\n");
        return 1;
    }
    printf("socket created\n");

    if (bind(sockfd, (struct sockaddr *) &server_addr, sizeof(server_addr)) < 0) {
        printf("couldn't bind socket\n");
        return 1;
    }
    printf("bind at port 3000\n");

    if (listen(sockfd, MAX_CLIENTS) < 0) {
        printf("couldn't listen to socket\n");
```



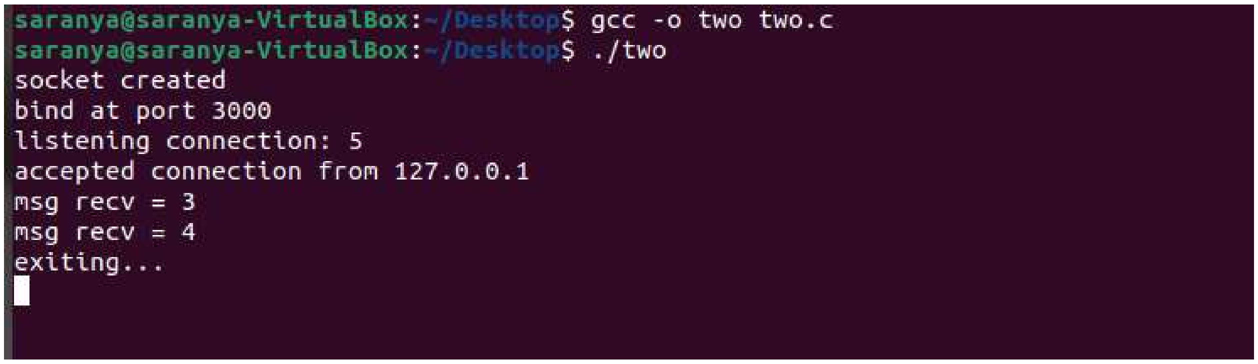
```c

    if (listen(sockfd, MAX_CLIENTS) < 0) {
        printf("couldn't listen to socket\n");
```

```c
        return 1;
    }
    printf("listening connection: %d\n", MAX_CLIENTS);

    struct sockaddr_in client_addr;
    int client_addr_size = sizeof(client_addr);

    while (1) {
        int client_sockfd;
        if ((client_sockfd = accept(sockfd, (struct sockaddr *) &client_addr, &client_addr_size)) < 0) {
            printf("couldn't accept connection\n");
            return 1;
        }
        printf("accepted connection from %s\n", inet_ntoa(client_addr.sin_addr));

        char msg[100];
        char card_number[20];
        char pin[5];
        recv(client_sockfd, msg, sizeof(msg), 0);
        sscanf(msg, "%s %s", card_number, pin);

        if (authenticate(card_number, pin)) {
            send(client_sockfd, "Authenticated", sizeof("Authenticated"), 0);
            while (1) {
                recv(client_sockfd, msg, sizeof(msg), 0);
                printf("msg recv = %s\n", msg);

                if (strcmp(msg, "4") == 0) {
                    printf("exiting...\n");
                    break;
                }
                send(client_sockfd, msg, sizeof(msg), 0);
            }
        } else {
            send(client_sockfd, "Authentication failed", sizeof("Authentication failed"), 0);
        close(client_sockfd);
    }

```

Output



```
saranya@saranya-VirtualBox:~/Desktop$ ./one
socket created
connected to the server
Enter card number: 1234567890123456
Enter 4 digit PIN: 1234
Authentication result: Authenticated
Options:
1. Deposit
2. Withdrawal
3. Check Balance
4. Exit
Enter option: 3
The balance is 1000000
Options:
1. Deposit
2. Withdrawal
3. Check Balance
4. Exit
Enter option: 4
Exiting...
saranya@saranya-VirtualBox:~/Desktop$
```



```
saranya@saranya-VirtualBox:~/Desktop$ gcc -o two two.c
saranya@saranya-VirtualBox:~/Desktop$ ./two
socket created
bind at port 3000
listening connection: 5
accepted connection from 127.0.0.1
msg recv = 3
msg recv = 4
exiting...
```