main.c

Share Run

Output

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define PORT 8080
#define BUFFER_SIZE 256

int main() {
    int sock;
    struct sockaddr_in serv_addr;
    char buffer[BUFFER_SIZE] = {0};

    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("Socket creation error");
        return -1;
    }

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT); // Convert port to network byte order
    serv_addr.sin_addr.s_addr = INADDR_ANY; // For simplicity, use any address

    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
        perror("Connection failed");
        return -1;
    }

    int valread = read(sock, buffer, BUFFER_SIZE);
    if (valread < 0) {
```

```
/tmp/wQDo08Tvf9.o
Connection failed: Connection refused


=== Code Exited With Errors ===
```

main.c

Share Run

Output

```c
#include <arpa/inet.h> // inet_addr()
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h> // bzero()
#include <sys/socket.h>
#include <unistd.h> // read(), write(), close()
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
void func(int sockfd)
{
    char buff[MAX];
    int n;
    for (;;) {
        bzero(buff, sizeof(buff));
        printf("Enter the string : ");
        n = 0;
        while ((buff[n++] = getchar()) != '\n')
            ;
        write(sockfd, buff, sizeof(buff));
        bzero(buff, sizeof(buff));
        read(sockfd, buff, sizeof(buff));
        printf("From Server : %s", buff);
        if ((strncmp(buff, "exit", 4)) == 0) {
            printf("Client Exit...\n");
            break;
        }
    }
}
```

```
/tmp/Q4roFsWnJ6.o
Socket successfully created..
connection with the server failed...


=== Code Execution Successful ===
```

main.c

Share | Run | Output

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

#define WINDOW_SIZE 4
#define TOTAL_PACKETS 10

void sendPackets(int windowSize) {
    int packetsSent = 0;
    while (packetsSent < TOTAL_PACKETS) {
        printf("Sending packets: ");
        for (int i = 0; i < windowSize && packetsSent < TOTAL_PACKETS; i++) {
            printf("%d ", packetsSent + 1);
            packetsSent++;
        }
        printf("\n");
        printf("Acknowledgment received for packets up to %d\n", packetsSent);
    }
}

int main() {
    sendPackets(WINDOW_SIZE);
    return 0;
}
```

```
/tmp/Zdus3ensWN.o
Sending packets: 1 2 3 4
Acknowledgment received for packets up to 4
Sending packets: 5 6 7 8
Acknowledgment received for packets up to 8
Sending packets: 9 10
Acknowledgment received for packets up to 10


=== Code Execution Successful ===
```

main.c

Share | Run | Output

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#include <netinet/if_ether.h>
#include <sys/socket.h>
#include <unistd.h>

#define CACHE_SIZE 10

typedef struct {
    char ip[INET_ADDRSTRLEN];
    unsigned char mac[ETH_ALEN];
} arp_cache_entry;

arp_cache_entry arp_cache[CACHE_SIZE];

void add_to_cache(const char *ip, unsigned char *mac) {
    for (int i = 0; i < CACHE_SIZE; i++) {
        if (strlen(arp_cache[i].ip) == 0) {
            strcpy(arp_cache[i].ip, ip);
            memcpy(arp_cache[i].mac, mac, ETH_ALEN);
            break;
        }
    }
}

void print_cache() {
    for (int i = 0; i < CACHE_SIZE; i++) {
        if (strlen(arp_cache[i].ip) > 0) {
            printf("IP: %s, MAC: %02x:%02x:%02x:%02x:%02x:%02x\n",
```

```
/tmp/R5kOOAdw6e.o
IP: 192.168.1.1, MAC: 00:11:22:33:44


=== Code Execution Successful ===
```

main.c

Share Run Output

```c
#include <arpa/inet.h> // inet_addr()
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h> // bzero()
#include <sys/socket.h>
#include <unistd.h> // read(), write(), close()
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
void func(int sockfd)
{
    char buff[MAX];
    int n;
    for (;;) {
        bzero(buff, sizeof(buff));
        printf("Enter the string : ");
        n = 0;
        while ((buff[n++] = getchar()) != '\n')
            ;
        write(sockfd, buff, sizeof(buff));
        bzero(buff, sizeof(buff));
        read(sockfd, buff, sizeof(buff));
        printf("From Server : %s", buff);
        if ((strncmp(buff, "exit", 4)) == 0) {
            printf("Client Exit...\n");
            break;
        }
    }
}
```

```
/tmp/vz0jF24a65.o
Socket successfully created..
connection with the server failed...


=== Code Execution Successful ===
```

main.c

Share Run Output

```c
#include <stdio.h>
#include <string.h>

#define POLYNOMIAL 0x07

unsigned char compute_crc(const char *data) {
    unsigned char crc = 0; // Initial CRC value
    size_t len = strlen(data);

    for (size_t i = 0; i < len; i++) {
        crc ^= (data[i] & 0xFF);

        for (int j = 0; j < 8; j++) {
            if (crc & 0x80) {
                crc = (crc << 1) ^ POLYNOMIAL;
            } else {
                crc <<= 1;
            }
        }
    }
    return crc;
}

void simulate_error(char *data) {
    size_t len = strlen(data);
    if (len > 0) {
        data[0] = (data[0] ^ 0x01);
    }
}

int check_crc(const char *data, unsigned char crc) {
```

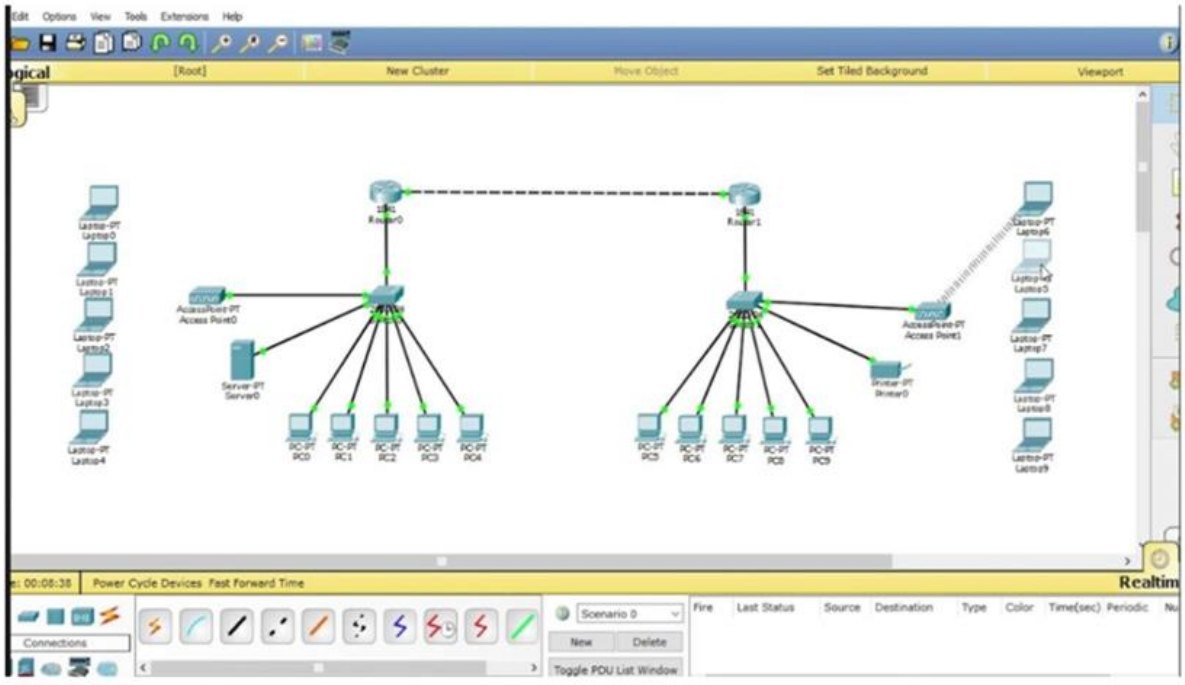
```
/tmp/OLJRS3Vdad.o
ERROR!
Original Data: Hello, World!
Computed CRC: 0x87
Data after error: Iello, World!
Error detected in data!


=== Code Execution Successful ===
```

main.c

Share | Run | Output

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define SERVER_IP "127.0.0.1"
#define PORT 5353
#define DOMAIN "google.com"
#define BUFFER_SIZE 512

int main() {
    int sockfd;
    struct sockaddr_in server_addr;
    char buffer[BUFFER_SIZE];

    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    inet_pton(AF_INET, SERVER_IP, &server_addr.sin_addr);

    sendto(sockfd, DOMAIN, strlen(DOMAIN), 0, (struct sockaddr *)&server_addr, sizeof
        (server_addr));
    printf("Sent query for: %s\n", DOMAIN);
```

```
/tmp/LUGnMXirYI.o
Sent query for: google.com
```



main.c

Share Run

Output

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#define PORT 5353
#define BUFFER_SIZE 512
#define DOMAIN "example.com"
#define IP_ADDRESS "93.184.216.34"

int main() {
    int sockfd;
    struct sockaddr_in server_addr, client_addr;
    socklen_t addr_len = sizeof(client_addr);
    char buffer[BUFFER_SIZE];

    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(PORT);

    if (bind(sockfd, (const struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        perror("Bind failed");
        exit(EXIT_FAILURE);
    }

```

```
/tmp/CbxB7pyn3c.o
DNS Server is listening on port 5353
```

main.c

Share Run

Output

```c
#include <stdio.h>
#include <stdlib.h>
#include <netdb.h>
#include <arpa/inet.h>

int main() {
    char *hostname = "www.example.com";
    struct hostent *host_info;
    struct in_addr **address_list;

    host_info = gethostbyname(hostname);

    if (host_info == NULL) {
        printf("Error: Could not resolve hostname.\n");
        return 1;
    }

    address_list = (struct in_addr **) host_info->h_addr_list;

    printf("IP Address: %s\n", inet_ntoa(*address_list[0]));

    return 0;
}
```

```
/tmp/TseKZkR01V.o
IP Address: 93.184.215.14


=== Code Execution Successful ===
```