POST LAB – 14

CLIENT SIDE

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

using namespace std;

int main(int argc, char** argv) {
    int client_sock;
    int error;

    client_sock = socket(PF_INET, SOCK_STREAM, 0);
    if (client_sock == -1) {
        perror("Socket creation failed.");
        return 1;
    }

    sockaddr_in server_address;
    memset(&server_address, 0, sizeof(server_address));
    server_address.sin_addr.s_addr = inet_addr("127.0.0.1");
    server_address.sin_family = PF_INET;
    server_address.sin_port = htons(8088);
```

```
    error = connect(client_sock, (struct sockaddr *)&server_address,
sizeof(server_address));
    if (error == -1) {
        perror("Failed to connect.");
        return 1;
    }

    int user_choice = 0;

    cout << "\t 1. Square" << endl;
    cout << "\t 2. Cube" << endl;

    cout << endl;

    cout << "Please Select Your Choice: ";
    cin >> user_choice;

    cout << endl;

    int32_t number;
    cout << "Enter an integer: ";
    cin >> number;

    if(user_choice == 1){
        int32_t sq_no = 6;
        char* str = "square";
        write(client_sock, &sq_no, sizeof(sq_no));
        write(client_sock, &str, sizeof(str));
        write(client_sock, &number, sizeof(number));

        int32_t squared_number;
        read(client_sock, &squared_number, sizeof(squared_number));
```

```
        cout << "The square of " << number << " is " << squared_number <<
endl;
    }

    else if(user_choice == 2){
        int32_t sq_no = 4;
        char* str = "cube";
        write(client_sock, &sq_no, sizeof(sq_no));
        write(client_sock, &str, sizeof(str));
        write(client_sock, &number, sizeof(number));

        int32_t cubed_number;
        read(client_sock, &cubed_number, sizeof(cubed_number));
        cout << "The cube of " << number << " is " << cubed_number <<
endl;
    }

    else{
        cout << "Invalid Choice" << endl;
    }

    close(client_sock);
    return 0;
}
```

SEVER SIDE

```
#include <stdlib.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>
```

```
#include <iostream>
#include <stdio.h>

using namespace std;

int main(int argc, char** argv) {
    int server_sock, client_sock;
    int error;

    server_sock = socket(PF_INET, SOCK_STREAM, 0);
    if (server_sock == -1) {
        perror("Socket creation failed.");
        return 1;
    }

    sockaddr_in server_address;
    memset(&server_address, '\0', sizeof(server_address));
    server_address.sin_addr.s_addr = inet_addr("127.0.0.1");
    server_address.sin_family = PF_INET;
    server_address.sin_port = htons(8088);

    error = bind(server_sock, (struct sockaddr *)&server_address,
sizeof(server_address));
    if (error == -1) {
        perror("Bind failed.");
        return 1;
    }

    error = listen(server_sock, 10);
    if (error == -1) {
        perror("Listen failed.");
        return 1;
```

```
    }

    cerr << "Now listening for connections...\n";

    while (true) {
        sockaddr_in client_address;
        socklen_t client_addr_len = sizeof(client_address);
        client_sock = accept(server_sock, (struct sockaddr
*)&client_address, &client_addr_len);
        if (client_sock == -1) {
            perror("Accept failed.");
            return 1;
        }

        cerr << "Connection request received.\n";

        int32_t size_str;
        read(client_sock, &size_str, sizeof(size_str));
        cout << size_str << endl;

        char* user_str;
        read(client_sock, &user_str, sizeof(user_str));

        int32_t received_int;
        read(client_sock, &received_int, sizeof(received_int));

        if(size_str == 6 && user_str == "square"){
            int32_t squared_int = received_int * received_int;
            write(client_sock, &squared_int, sizeof(squared_int));
        }

        else if(size_str == 4 && user_str == "cube"){
```

```
            int32_t cubed_int = received_int * received_int *
received_int;
            write(client_sock, &cubed_int, sizeof(cubed_int));
        }

    }


    close(client_sock);
    close(server_sock);
    return 0;
}
```

OUTPUT