SWITCH CASES

Server Code:

```
#include <iostream>
#include <unistd.h>
#include <cstring>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <bits/stdc++.h>
using namespace std;
#define PORT 8080
#define BUFFER_SIZE 50
int main()
{
    int server_fd, new_socket;
    char buffer[BUFFER_SIZE] = {0};
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
    {
        cerr << "Socket creation failed" << endl;
        exit(EXIT_FAILURE);
    }
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
                   sizeof(opt)))
    {
        cerr << "setsockopt failed" << endl;
        close(server_fd);
        exit(EXIT_FAILURE);
    }

    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0)
    {
        cerr << "Bind failed" << endl;
        close(server_fd);
        exit(EXIT_FAILURE);
    }
    if (listen(server_fd, 3) < 0)
    {
        cerr << "Listen failed" << endl;
        close(server_fd);
        exit(EXIT_FAILURE);
    }
    if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
                             (socklen_t *)&addrlen)) < 0)
    {
        cerr << "Accept failed" << endl;
        close(server_fd);
        exit(EXIT_FAILURE);
    }
    int valread = read(new_socket, buffer, BUFFER_SIZE);
    cout << "Client: " << buffer << endl;
    char message[BUFFER_SIZE] = {0};
    int j = 0;
    for (char i : buffer)
    {
        if (i >= 'a' && i <= 'z')
        {
            message[j++] = toupper(i);
        }
        else if (i >= 'A' && i <= 'Z')
        {
            message[j++] = tolower(i);
        }
        else
        {
            message[j++] = i;
        }
    }
    send(new_socket, message, BUFFER_SIZE, 0);
    cout << "Message sent to client" << endl;
    close(new_socket);
    close(server_fd);
    return 0;
}
```

Client Code:

```
#include <iostream>
#include <unistd.h>
#include <cstring>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
using namespace std;
#define PORT 8080
#define BUFFER_SIZE 50
int main()
{
    int sock = 0;
    struct sockaddr_in serv_addr;
    char buffer[BUFFER_SIZE] = {0};
    string message;
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        cerr << "Socket creation failed" << endl;
        exit(EXIT_FAILURE);
    }

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0)
    {
        cerr << "Invalid address/ Address not supported" << endl;
        close(sock);
        exit(EXIT_FAILURE);
    }
    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
    {
        cerr << "Connection Failed" << endl;
        close(sock);
        exit(EXIT_FAILURE);
    }
    while (1)
    {
        cout << "Enter the string: ";
        cin >> buffer;
        send(sock, buffer, BUFFER_SIZE, 0);
        cout << "Message sent to server" << endl;
        int valread = read(sock, buffer, BUFFER_SIZE);
        cout << sizeof(buffer) << endl;
        if (buffer[0] == 0)
            cout << 0 << endl;
        else
            cout << "Server: " << buffer << endl;
    }
    close(sock);
    return 0;
}
```

OUTPUT :

```
rana:~/Desktop/lab$ g++ 1s.cpp
rana:~/Desktop/lab$./a.out
Client: RaNaDEEsh
Message sent to client


rana:~/Desktop/lab$ g++ 1c.cpp
rana:~/Desktop/lab$./a.out
Enter the string : RaNaDEEsh
Message sent to server
Server : rAnAdeeSH
Enter the string :
```

POSTFIX EVALUATION

Server Code :

```
#include <iostream>
#include <unistd.h>
#include <cstring>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <bits/stdc++.h>
using namespace std;
#define PORT 8080
#define BUFFER_SIZE 50
int main()
{
    int server_fd, new_socket;
    char buffer[BUFFER_SIZE] = {0};
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
    {
        cerr << "Socket creation failed" << endl;
        exit(EXIT_FAILURE);
    }
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
                   sizeof(opt)))
    {
        cerr << "setsockopt failed" << endl;
        close(server_fd);
        exit(EXIT_FAILURE);
    }

    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0)
    {
        cerr << "Bind failed" << endl;
        close(server_fd);
        exit(EXIT_FAILURE);
    }
    if (listen(server_fd, 3) < 0)
    {
        cerr << "Listen failed" << endl;
        close(server_fd);
        exit(EXIT_FAILURE);
    }

    if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
                             (socklen_t *)&addrlen)) < 0)
    {
        cerr << "Accept failed" << endl;
        close(server_fd);
        exit(EXIT_FAILURE);
    }
```

```
while (1)
{
    int valread = read(new_socket, buffer, BUFFER_SIZE);
    cout << "Client: " << buffer << endl;
    stack<int> s;
    for (char i : buffer)
    {
        if (i >= '0' && i <= '9')
        {
            s.push(i - '0');
        }
        else if (i == '+')
        {
            int first = 0, second = 0;
            if (!s.empty())
                second = s.top();
            s.pop();
            if (!s.empty())
                first = s.top();
            s.pop();
            s.push(first + second);
        }
        else if (i == '-')
        {
            int first = 0, second = 0;
            if (!s.empty())
                second = s.top();
            s.pop();
            if (!s.empty())
                first = s.top();
            s.pop();
            s.push(first - second);
        }
        else if (i == '*')
        {
            int first = 0, second = 0;
            if (!s.empty())
                second = s.top();
            s.pop();
            if (!s.empty())
                first = s.top();
            s.pop();
            s.push(first * second);
        }
        else if (i == '/')
        {
            int first = 0, second = 0;
            if (!s.empty())
                second = s.top();
            s.pop();
            if (!s.empty())
                first = s.top();
            s.pop();
            s.push(first / second);
        }
    }
```

```
        int d = s.top();
        char message[BUFFER_SIZE] = {0};
        int j = log10(d);
        while (d)
        {
            message[j--] = (d % 10) + '0';
            d /= 10;
        }
        send(new_socket, &message, BUFFER_SIZE, 0);
        cout << "Message sent to client" << endl;
    }
    close(new_socket);
    close(server_fd);
    return 0;
}
```

Client Code :

```
#include <iostream>
#include <unistd.h>
#include <cstring>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
using namespace std;
#define PORT 8080
#define BUFFER_SIZE 50
int main()
{
    int sock = 0;
    struct sockaddr_in serv_addr;
    char buffer[BUFFER_SIZE] = {0};
    string message;
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        cerr << "Socket creation failed" << endl;
        exit(EXIT_FAILURE);
    }

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0)
    {
        cerr << "Invalid address/ Address not supported" << endl;
        close(sock);
        exit(EXIT_FAILURE);
    }
    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
    {
        cerr << "Connection Failed" << endl;
        close(sock);
        exit(EXIT_FAILURE);
    }
    while (1)
    {
        cout << "Enter the string: ";
        cin >> buffer;

        send(sock, buffer, BUFFER_SIZE, 0);
        cout << "Message sent to server" << endl;

        int valread = read(sock, buffer, BUFFER_SIZE);
        cout << sizeof(buffer) << endl;

        if (buffer[0] == 0)
            cout << 0 << endl;
        else
            cout << "Server: " << buffer << endl;
    }

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

OUTPUT :

```
rana:~/Desktop/lab$ g++ 2s.cpp

rana:~/Desktop/lab$./a.out

Client: 22+3*

Message sent to client


rana:~/Desktop/lab$ g++ 2c.cpp

rana:~/Desktop/lab$./a.out

Enter the string : 22+3*

Message sent to server

Server : 12

Enter the string :
```

STRING EXPRESSION EVALUATION

Server Code:

```
#include <iostream>
#include <unistd.h>
#include <cstring>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <bits/stdc++.h>

using namespace std;
#define PORT 8080
#define BUFFER_SIZE 50

int main()
{
    int server_fd, new_socket;

    char buffer[BUFFER_SIZE] = {0};

    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);

    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
    {
        cerr << "Socket creation failed" << endl;
        exit(EXIT_FAILURE);
    }

    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
        sizeof(opt))){
        cerr << "setsockopt failed" << endl;
        close(server_fd);
        exit(EXIT_FAILURE);
    }

    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0)
    {
        cerr << "Bind failed" << endl;
        close(server_fd);
        exit(EXIT_FAILURE);
    }
    if (listen(server_fd, 3) < 0)
    {
        cerr << "Listen failed" << endl;
        close(server_fd);
        exit(EXIT_FAILURE);
    }

    if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
                             (socklen_t *)&addrlen)) < 0){
        cerr << "Accept failed" << endl;
        close(server_fd);
        exit(EXIT_FAILURE);
    }
    while (1)
    {
        int valread = read(new_socket, buffer, BUFFER_SIZE);
        cout << "Client: " << buffer << endl;
        int first = 0, turn = 0, second = 0;
        char op;
        for (char i : buffer)
        {
            if (i >= '0' && i <= '9' && first == 0)
                first = i - '0';
            else if (i >= '0' && i <= '9' && second == 0)
                second = i - '0';
            else if (i == '+' || i == '-' || i == '*' || i == '/')
                op = i;
            else
                continue;
        }
        int d = 0;
        if (op == '+')
            d = first + second;
        else if (op == '-')
            d = first - second;
        else if (op == '*')
            d = first * second;
        else if (op == '/')
            d = first / second;
        char message[BUFFER_SIZE] = {0};
        int j = log10(d);
        while (d)
        {
            message[j--] = (d % 10) + '0';
            d /= 10;
        }
        send(new_socket, &message, BUFFER_SIZE, 0);
        cout << "Message sent to client" << endl;
    }

    close(new_socket);
    close(server_fd);

    return 0;
}
```

Client Code:

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

using namespace std;
#define PORT 8080
#define BUFFER_SIZE 50

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

    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        cerr << "Socket creation failed" << endl;
        exit(EXIT_FAILURE);
    }

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0)
    {
        cerr << "Invalid address/ Address not supported" << endl;
        close(sock);
        exit(EXIT_FAILURE);
    }

    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
    {
        cerr << "Connection Failed" << endl;
        close(sock);
        exit(EXIT_FAILURE);
    }

    while (1)
    {
        cout << "Enter the string: ";
        cin >> buffer;

        send(sock, buffer, BUFFER_SIZE, 0);

        cout << "Message sent to server" << endl;


        int valread = read(sock, buffer, BUFFER_SIZE);
        cout << sizeof(buffer) << endl;
        if (buffer[0] == 0)
            cout << 0 << endl;
        else
            cout << "Server: " << buffer << endl;
    }
    close(sock);
    return 0;
}
```

OUTPUT :

```
rana:~/Desktop/lab$ g++ 3s.cpp

rana:~/Desktop/lab$./a.out

Client: 4*2

Message sent to client


rana:~/Desktop/lab$ g++ 3c.cpp

rana:~/Desktop/lab$./a.out

Enter the string : 4*2

Message sent to server

Server : 8

Enter the string :
```