ASSIGNMENT -> TCP SOCKET

1.a

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

#define MAX_STR_LEN 100

void reverseString(char *str)
    { int len = strlen(str);
    for (int i = 0; i < len / 2; i++)
        { char temp = str[i];
        str[i] = str[len - i - 1];
        str[len - i - 1] = temp;
    }
}

int main(int argc, char *argv[])
    { if (argc != 2) {
        fprintf(stderr, "Usage: %s <port>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    int port = atoi(argv[1]);
```

```
    int sockfd, clientfd;
    struct sockaddr_in serverAddr, clientAddr;
    socklen_t addrLen = sizeof(clientAddr);

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

    memset(&serverAddr, 0, sizeof(serverAddr));
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_addr.s_addr = htonl(INADDR_ANY);
    serverAddr.sin_port = htons(port);

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

    if (listen(sockfd, 5) < 0)
        {          perror("Listen
        failed");
        exit(EXIT_FAILURE);
    }

    printf("Server listening on port %d\n", port);

    while (1) {
        clientfd = accept(sockfd, (struct sockaddr *)&clientAddr, &addrLen);
        if (clientfd < 0)
            { perror("Accept failed");
            exit(EXIT_FAILURE);
        }

        char buffer[MAX_STR_LEN];
        ssize_t bytesRead = read(clientfd, buffer, sizeof(buffer));
        if (bytesRead < 0) {
            perror("Read failed");
            exit(EXIT_FAILURE);
        }

        buffer[bytesRead] = '\0';
        printf("Received from client: %s\n", buffer);

        reverseString(buffer);
```

```
        ssize_t bytesSent = write(clientfd, buffer, strlen(buffer));
        if (bytesSent < 0) {
            perror("Write failed");
            exit(EXIT_FAILURE);
        }

        printf("Sending to client: %s\n", buffer);

        close(clientfd);
    }

    close(sockfd);

    return 0;
}
```

1.b

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

#define MAX_STR_LEN 100

int main(int argc, char *argv[])
    { if (argc != 3) {
        fprintf(stderr, "Usage: %s <server_ip> <port>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    char *serverIp = argv[1];
    int port = atoi(argv[2]);

    int sockfd;
    struct sockaddr_in serverAddr;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0) {
        perror("Socket creation failed");
```

```
        exit(EXIT_FAILURE);
    }

    memset(&serverAddr, 0, sizeof(serverAddr));
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_addr.s_addr = inet_addr(serverIp);
    serverAddr.sin_port = htons(port);

    if (connect(sockfd, (struct sockaddr *)&serverAddr, sizeof(serverAddr)) <
0) {
        perror("Connection failed");
        exit(EXIT_FAILURE);
    }

    char inputStr[MAX_STR_LEN];
    printf("Enter a string to reverse: ");
    fgets(inputStr, sizeof(inputStr), stdin);

    ssize_t bytesSent = write(sockfd, inputStr, strlen(inputStr));
    if (bytesSent < 0) {
        perror("Write failed");
        exit(EXIT_FAILURE);
    }

    char buffer[MAX_STR_LEN];
    ssize_t bytesRead = read(sockfd, buffer, sizeof(buffer));
    if (bytesRead < 0) {
        perror("Read failed");
        exit(EXIT_FAILURE);
    }

    buffer[bytesRead] = '\0';
    printf("Reversed string received from server: %s\n", buffer);

    close(sockfd);

    return 0;
}
```

2. a

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

#define MAX_EXPR_LEN 100

float evaluateExpression(char *expr)
    { float operand1, operand2;
    char operator;
    sscanf(expr, "%f %c %f", &operand1, &operator, &operand2);

    switch (operator)
        { case '+':
            return operand1 + operand2;
        case '-':
            return operand1 - operand2;
        case '*':
            return operand1 * operand2;
        case '/':
            if (operand2 != 0) {
                return operand1 / operand2;
            } else {
                fprintf(stderr, "Error: Division by zero\n");
                return 0;
            }
        default:
            fprintf(stderr, "Error: Invalid operator\n");
            return 0;
    }
}

int main(int argc, char *argv[])
    { if (argc != 2) {
        fprintf(stderr, "Usage: %s <port>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    int port = atoi(argv[1]);

    int sockfd, clientfd;
    struct sockaddr_in serverAddr, clientAddr;
    socklen_t addrLen = sizeof(clientAddr);

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0) {
        perror("Socket creation failed");
```

```
        exit(EXIT_FAILURE);
    }

    memset(&serverAddr, 0, sizeof(serverAddr));
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_addr.s_addr = htonl(INADDR_ANY);
    serverAddr.sin_port = htons(port);

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

    if (listen(sockfd, 5) < 0)
        {         perror("Listen
        failed");
        exit(EXIT_FAILURE);
    }

    printf("Server listening on port %d\n", port);

    while (1) {
        clientfd = accept(sockfd, (struct sockaddr *)&clientAddr, &addrLen);
        if (clientfd < 0)
            { perror("Accept failed");
            exit(EXIT_FAILURE);
        }

        printf("TCP client connected from %s on port %d\n",
inet_ntoa(clientAddr.sin_addr), ntohs(clientAddr.sin_port));

        char buffer[MAX_EXPR_LEN];
        ssize_t bytesRead = read(clientfd, buffer, sizeof(buffer));
        if (bytesRead < 0) {
            perror("Read failed");
            exit(EXIT_FAILURE);
        }

        buffer[bytesRead] = '\0';
        printf("Received from client: %s", buffer);

        float result = evaluateExpression(buffer);
        snprintf(buffer, sizeof(buffer), "%.2f", result);

        ssize_t bytesSent = write(clientfd, buffer, strlen(buffer));
        if (bytesSent < 0) {
            perror("Write failed");
            exit(EXIT_FAILURE);
```

```
        }

        printf("Sending to client: %s\n", buffer);

        close(clientfd);
    }

    close(sockfd);

    return 0;
}
```

2. b

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

#define MAX_EXPR_LEN 100

int main(int argc, char *argv[])
    { if (argc != 3) {
        fprintf(stderr, "Usage: %s <server_hostname> <port>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    char *serverHostname = argv[1];
    int port = atoi(argv[2]);

    int sockfd;
    struct sockaddr_in serverAddr;

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

```
    struct addrinfo hints, *server;
    memset(&hints, 0, sizeof(hints));
    hints.ai_family = AF_INET;
    hints.ai_socktype = SOCK_STREAM;

    int status = getaddrinfo(serverHostname, NULL, &hints, &server);
    if (status != 0) {
        fprintf(stderr, "Error: %s\n", gai_strerror(status));
        exit(EXIT_FAILURE);
    }

    memcpy(&serverAddr, server->ai_addr, sizeof(serverAddr));
    serverAddr.sin_port = htons(port);

    freeaddrinfo(server);

    if (connect(sockfd, (struct sockaddr *)&serverAddr, sizeof(serverAddr)) <
0) {
        perror("Connection failed");
        exit(EXIT_FAILURE);
    }

    printf("TCP client connected to %s on port %d\n", serverHostname, port);

    char inputExpr[MAX_EXPR_LEN];
    while (1) {
        printf("Enter an expression in the following format (operand1 operator
operand2): ");
        fgets(inputExpr, sizeof(inputExpr), stdin);
        if (strcmp(inputExpr, "-1\n") == 0) {
            break;
        }

        ssize_t bytesSent = write(sockfd, inputExpr, strlen(inputExpr));
        if (bytesSent < 0) {
            perror("Write failed");
            exit(EXIT_FAILURE);
        }

        char buffer[MAX_EXPR_LEN];
        ssize_t bytesRead = read(sockfd, buffer, sizeof(buffer));
        if (bytesRead < 0) {
            perror("Read failed");
            exit(EXIT_FAILURE);
        }

        buffer[bytesRead] = '\0';
        printf("ANS: %s\n", buffer);
```

```
    }

    close(sockfd);

    return 0;
}
```