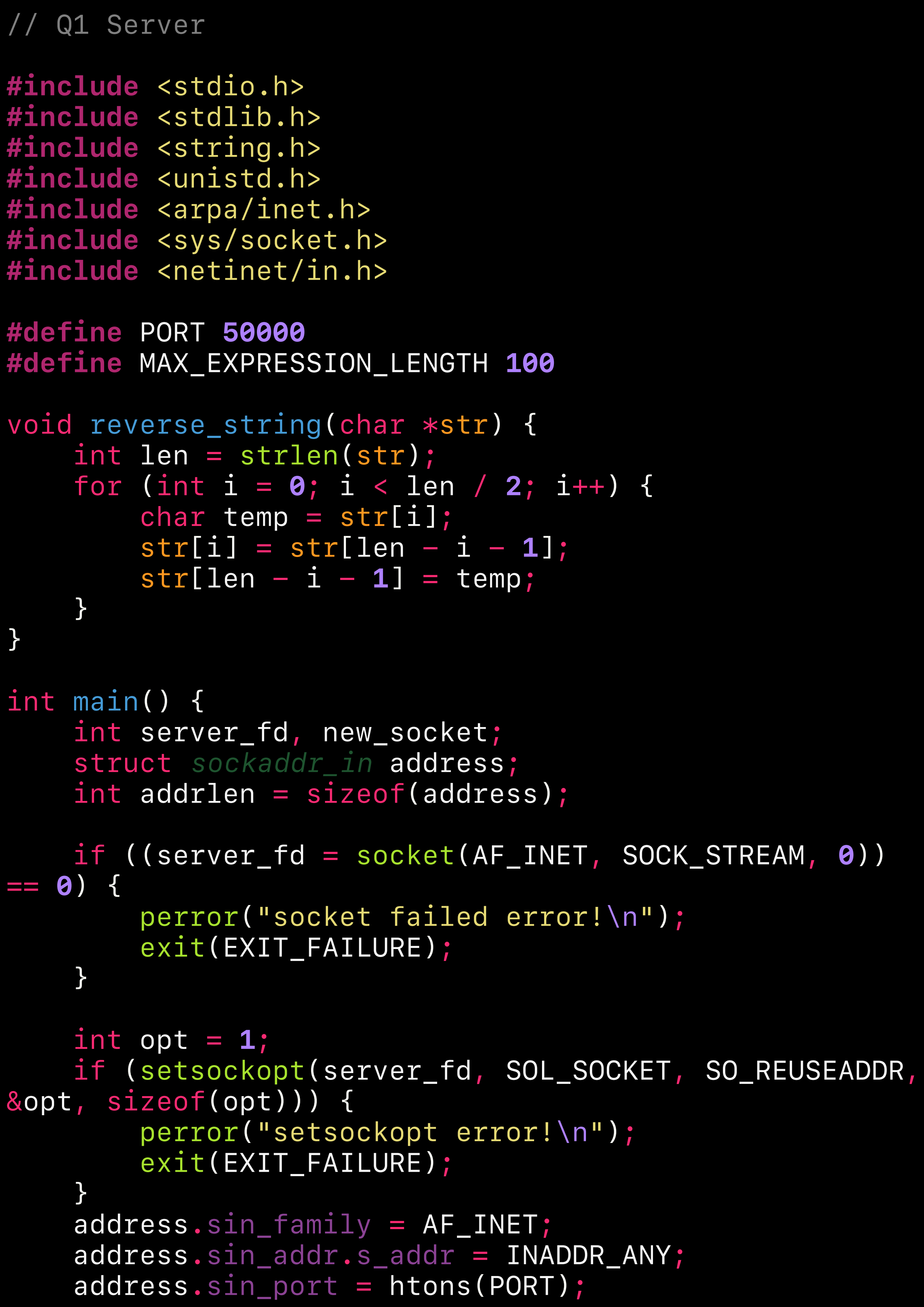
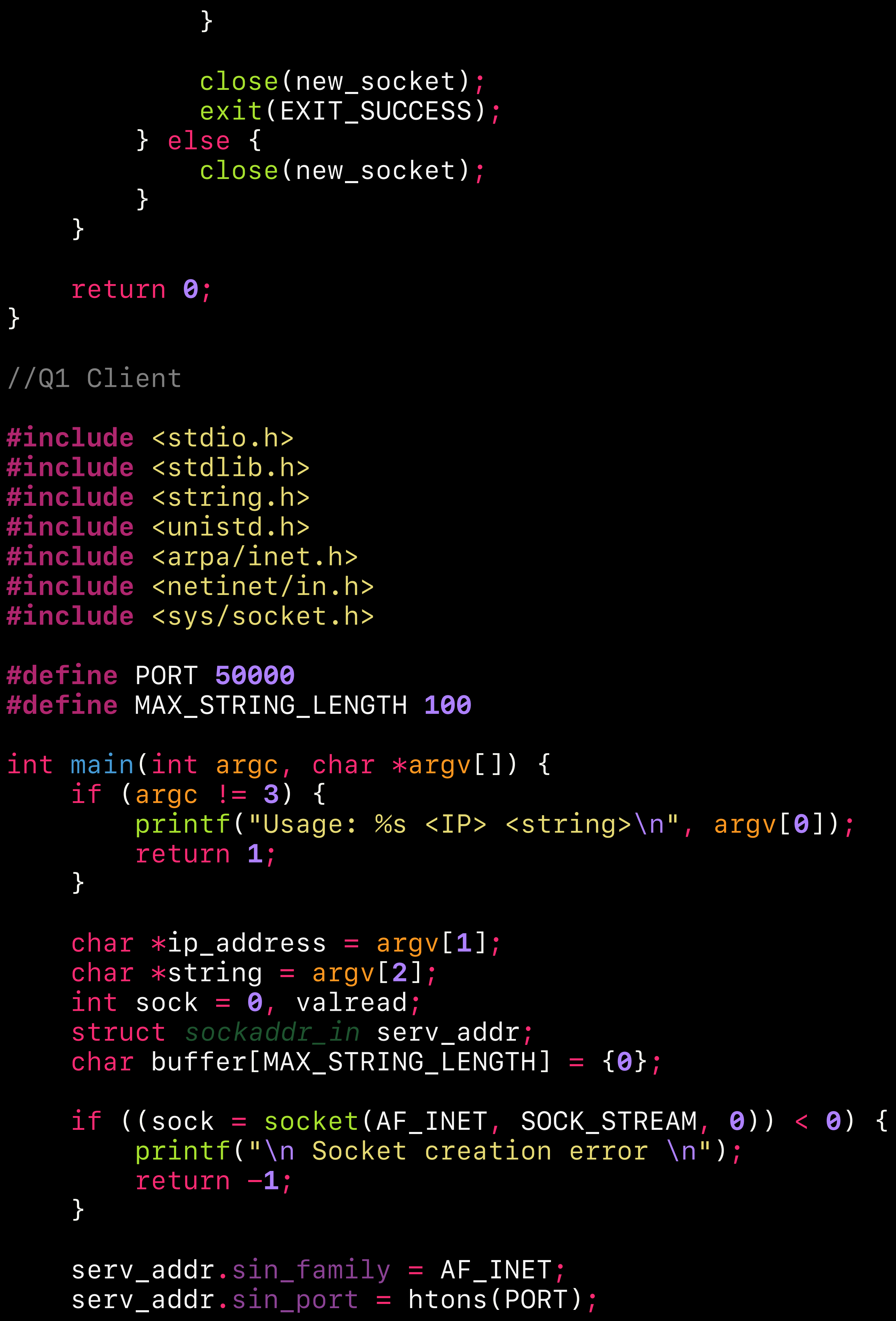
ASSIGNMENT 11
25/04/24

NAME : SHRESTH SONKAR
REGNO : 20214272
GROUP : CS6D
TOPIC : NETWORK LAB
CODE : CS-16204

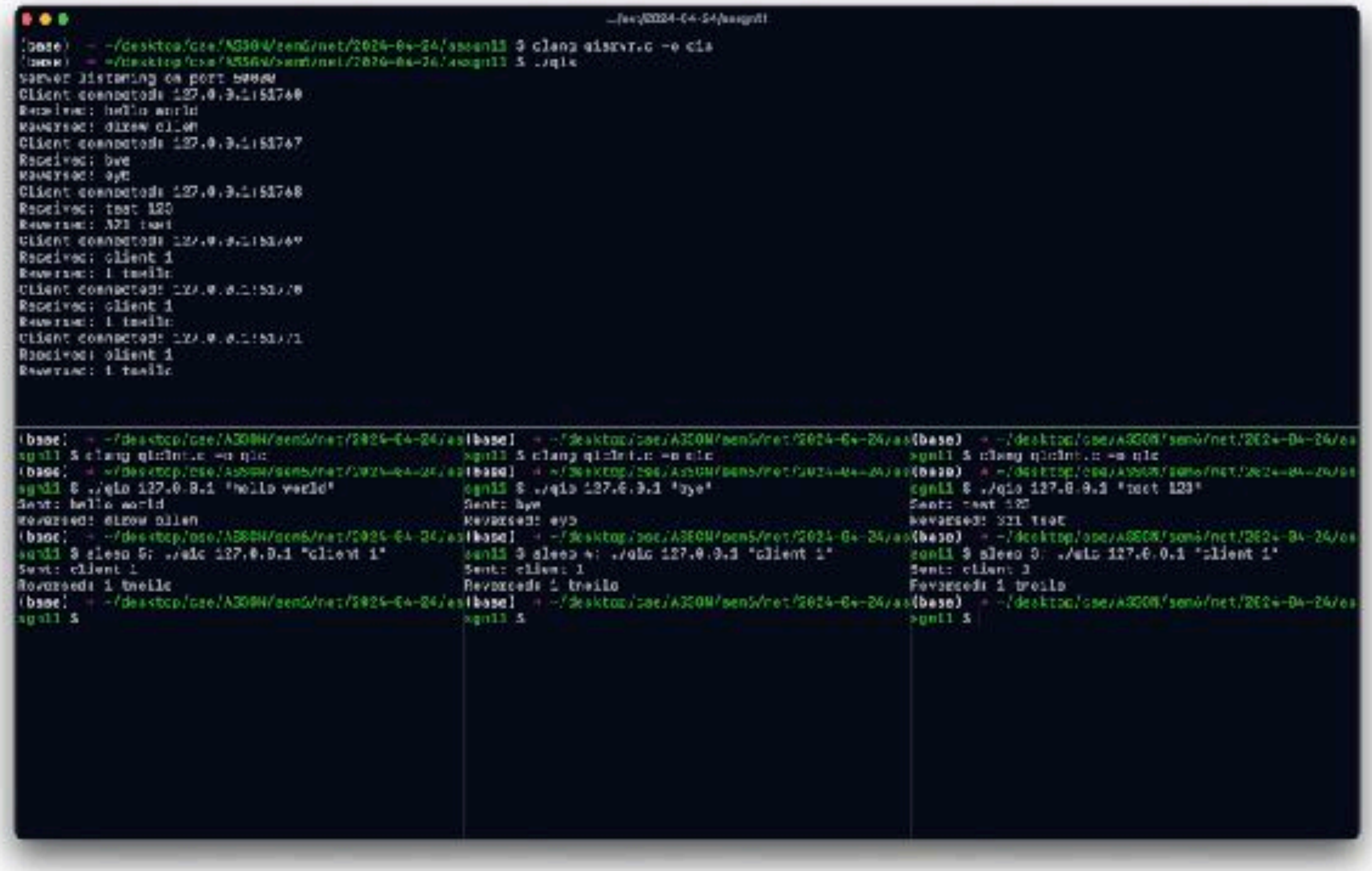


```c
// Q1 Server

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

#define PORT 50000
#define MAX_EXPRESSION_LENGTH 100

void reverse_string(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 server_fd, new_socket;
    struct sockaddr_in address;
    int addrlen = sizeof(address);

    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0))
== 0) {
        perror("socket failed error!\n");
        exit(EXIT_FAILURE);
    }

    int opt = 1;
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR,
&opt, sizeof(opt))) {
        perror("setsockopt error!\n");
        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) {
        perror("bind failed error!\n");
        exit(EXIT_FAILURE);
    }
    if (listen(server_fd, 3) < 0) {
        perror("listen error!\n");
        exit(EXIT_FAILURE);
    }

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

    while (1) {
        if ((new_socket = accept(server_fd, (struct
sockaddr *) &address, (socklen_t * ) & addrlen)) < 0) {
            perror("accept error!\n");
            exit(EXIT_FAILURE);
        }

        printf("Client connected: %s:%d\n",
inet_ntoa(address.sin_addr), ntohs(address.sin_port));
        pid_t pid = fork();

        if (pid < 0) {
            perror("fork error!\n");
            exit(EXIT_FAILURE);
        }

        if (pid == 0) {
            close(server_fd);

            char buffer[MAX_EXPRESSION_LENGTH] = {0};
            while (1) {
                int valread = read(new_socket, buffer,
MAX_EXPRESSION_LENGTH);
                if (valread <= 0) {
                    break;
                }

                printf("Received: %s\n", buffer);
                reverse_string(buffer);

                send(new_socket, buffer,
strlen(buffer), 0);
                printf("Reversed: %s\n", buffer);
```



```
            }

            close(new_socket);
            exit(EXIT_SUCCESS);
        } else {
            close(new_socket);
        }
    }

    return 0;
}

//Q1 Client

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

#define PORT 50000
#define MAX_STRING_LENGTH 100

int main(int argc, char *argv[]) {
    if (argc != 3) {
        printf("Usage: %s <IP> <string>\n", argv[0]);
        return 1;
    }

    char *ip_address = argv[1];
    char *string = argv[2];
    int sock = 0, valread;
    struct sockaddr_in serv_addr;
    char buffer[MAX_STRING_LENGTH] = {0};

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

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

```c
    if (inet_pton(AF_INET, ip_address,
&serv_addr.sin_addr) <= 0) {
        printf("\nInvalid address/ Address not
supported \n");
        return -1;
    }

    if (connect(sock, (struct sockaddr *)&serv_addr,
sizeof(serv_addr)) < 0) {
        printf("\nConnection Failed \n");
        return -1;
    }

    send(sock, string, strlen(string), 0);
    printf("Sent: %s\n", string);

    valread = read(sock, buffer, MAX_STRING_LENGTH);
    printf("Reversed: %s\n", buffer);
    return 0;
}
```



```
…/net/2024-04-24/assgn11

(base) ➜ ~/desktop/cse/A3304/sem5/net/2024-04-24/assgn11 $ clang q1server.c -o q1s
(base) ➜ ~/desktop/cse/A3304/sem5/net/2024-04-24/assgn11 $ ./q1s
Server listening on port 8080
Client connected: 127.0.0.1:51766
Received: hello world
Reversed: dlrow olleh
Client connected: 127.0.0.1:51767
Received: bye
Reversed: eyb
Client connected: 127.0.0.1:51768
Received: test 123
Reversed: 321 tset
Client connected: 127.0.0.1:51769
Received: client 1
Reversed: 1 tneilc
Client connected: 127.0.0.1:51770
Received: client 1
Reversed: 1 tneilc
Client connected: 127.0.0.1:51771
Received: client 1
Reversed: 1 tneilc
```

```
(base) ➜ ~/desktop/cse/A3304/sem5/net/2024-04-24/as
sgn11 $ clang q1client.c -o q1c
(base) ➜ ~/desktop/cse/A3304/sem5/net/2024-04-24/as
sgn11 $ ./q1c 127.0.0.1 "hello world"
Sent: hello world
Reversed: dlrow olleh
(base) ➜ ~/desktop/cse/A3304/sem5/net/2024-04-24/as
sgn11 $ sleep 5; ./q1c 127.0.0.1 "client 1"
Sent: client 1
Reversed: 1 tneilc
(base) ➜ ~/desktop/cse/A3304/sem5/net/2024-04-24/as
sgn11 $
```

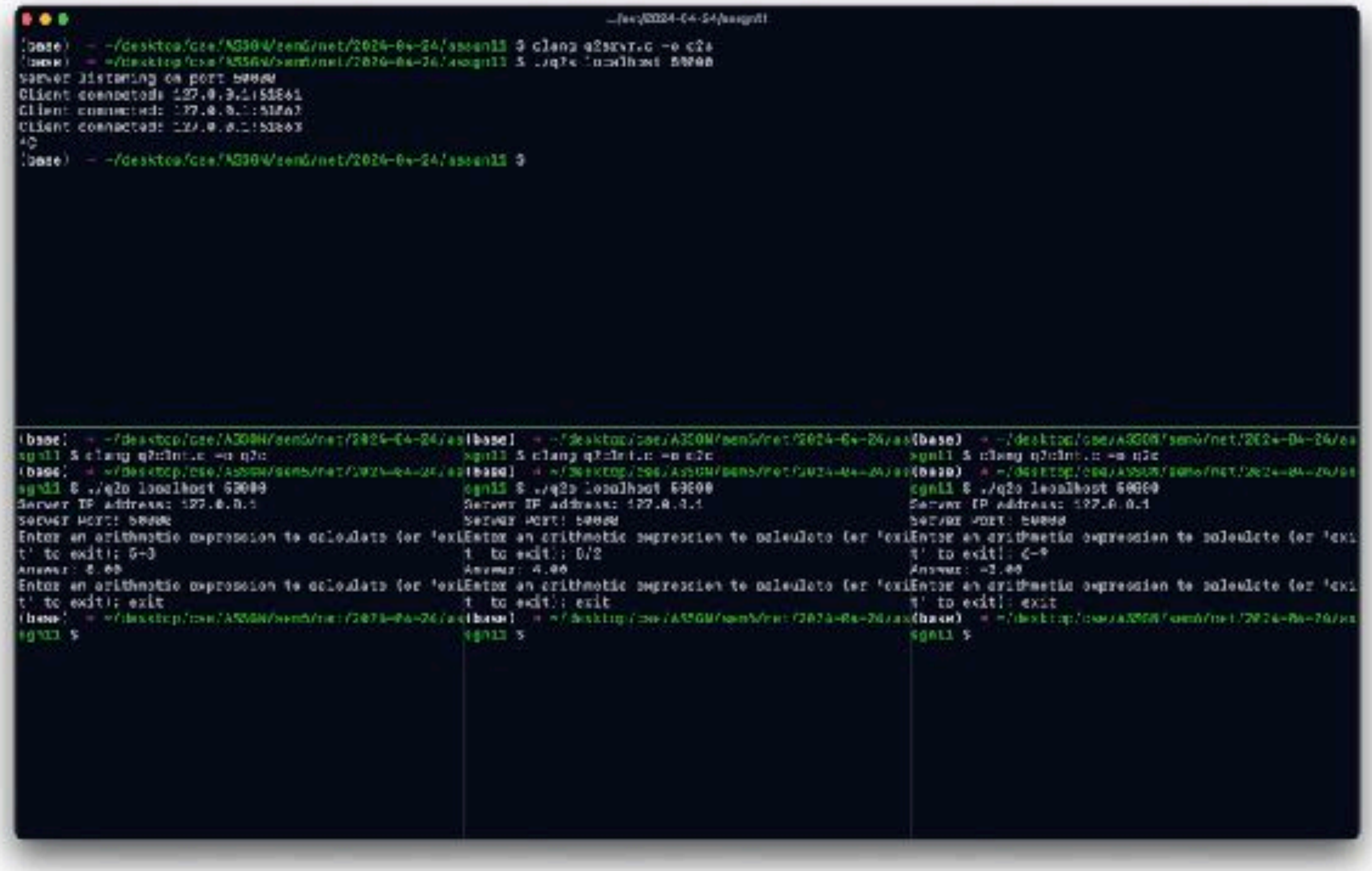
```
(base) ➜ ~/desktop/cse/A3304/sem5/net/2024-04-24/as
sgn11 $ clang q1client.c -o q1c
(base) ➜ ~/desktop/cse/A3304/sem5/net/2024-04-24/as
sgn11 $ ./q1c 127.0.0.1 "bye"
Sent: bye
Reversed: eyb
(base) ➜ ~/desktop/cse/A3304/sem5/net/2024-04-24/as
sgn11 $ sleep 4; ./q1c 127.0.0.1 "client 1"
Sent: client 1
Reversed: 1 tneilc
(base) ➜ ~/desktop/cse/A3304/sem5/net/2024-04-24/as
sgn11 $
```

```
(base) ➜ ~/desktop/cse/A3304/sem5/net/2024-04-24/as
sgn11 $ clang q1client.c -o q1c
(base) ➜ ~/desktop/cse/A3304/sem5/net/2024-04-24/as
sgn11 $ ./q1c 127.0.0.1 "test 123"
Sent: test 123
Reversed: 321 tset
(base) ➜ ~/desktop/cse/A3304/sem5/net/2024-04-24/as
sgn11 $ sleep 3; ./q1c 127.0.0.1 "client 1"
Sent: client 1
Reversed: 1 tneilc
(base) ➜ ~/desktop/cse/A3304/sem5/net/2024-04-24/as
sgn11 $
```

```c
// Q2 Server

#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <pthread.h>

#define PORT 50000
#define MAX_EXPRESSION_LENGTH 100

struct StackNode {
    int data;
    struct StackNode *next;
};

struct StackNode *newNode(int data) {
    struct StackNode *stackNode = (struct StackNode *)
malloc(sizeof(struct StackNode));
    stackNode->data = data;
    stackNode->next = NULL;
    return stackNode;
}

void push(struct StackNode **root, int data) {
    struct StackNode *stackNode = newNode(data);
    stackNode->next = *root;
    *root = stackNode;
}

int pop(struct StackNode **root) {
    if (*root == NULL) {
        fprintf(stderr, "Error: Stack underflow\n");
        exit(EXIT_FAILURE);
    }
    struct StackNode *temp = *root;
    *root = (*root)->next;
    int popped = temp->data;
    free(temp);
    return popped;
}
```

```
int peek(struct StackNode *root) {
    if (root == NULL) {
        fprintf(stderr, "Error: Stack empty\n");
        exit(EXIT_FAILURE);
    }
    return root->data;
}

int evaluate_expression(const char *expression) {
    struct StackNode *values = NULL;
    struct StackNode *ops = NULL;

    for (int i = 0; expression[i]; i++) {
        if (expression[i] == ' ')
            continue;

        if (isdigit(expression[i])) {
            int num = 0;
            while (isdigit(expression[i])) {
                num = num * 10 + (expression[i] - '0');
                i++;
            }
            i--;

            push(&values, num);
        } else if (expression[i] == '(') {
            push(&ops, expression[i]);
        } else if (expression[i] == ')') {
            while (peek(ops) != '(') {
                int val2 = pop(&values);
                int val1 = pop(&values);
                char op = pop(&ops);

                switch (op) {
                    case '+':
                        push(&values, val1 + val2);
                        break;
                    case '-':
                        push(&values, val1 - val2);
                        break;
                    case '*':
                        push(&values, val1 * val2);
                        break;
                    case '/':
                        push(&values, val1 / val2);
                        break;
                }
            }
            pop(&ops);
        } else {
            while (ops != NULL && peek(ops) != '(' &&
((expression[i] == '*' || expression[i] == '/') ||

(peek(ops) == '+' || peek(ops) == '-'))) {
                int val2 = pop(&values);
                int val1 = pop(&values);
                char op = pop(&ops);

                switch (op) {
                    case '+':
                        push(&values, val1 + val2);
                        break;
                    case '-':
                        push(&values, val1 - val2);
                        break;
                    case '*':
                        push(&values, val1 * val2);
                        break;
                    case '/':
                        push(&values, val1 / val2);
                        break;
                }
            }

            push(&ops, expression[i]);
        }
    }

    while (ops != NULL) {
        int val2 = pop(&values);
        int val1 = pop(&values);
        char op = pop(&ops);

        switch (op) {
            case '+':
                push(&values, val1 + val2);
                break;
            case '-':
                push(&values, val1 - val2);
                break;
            case '*':
                push(&values, val1 * val2);
                break;
            case '/':
                push(&values, val1 / val2);
                break;
        }
    }

    return pop(&values);
}


void *handle_client(void *arg) {
    int client_socket = *((int *) arg);
    char expression[MAX_EXPRESSION_LENGTH];
    char result[MAX_EXPRESSION_LENGTH];

    while (1) {
        int bytes_received = recv(client_socket,
expression, MAX_EXPRESSION_LENGTH, 0);
        if (bytes_received <= 0) {
            break;
        }

        expression[bytes_received] = '\0';
        double result_value =
evaluate_expression(expression);
        sprintf(result, "%.2lf", result_value);
        send(client_socket, result, strlen(result), 0);
        memset(expression, 0, strlen(expression));
    }

    close(client_socket);
    pthread_exit(NULL);
}

int main() {
    int server_fd, new_socket;
    struct sockaddr_in address;
    int addrlen = sizeof(address);

    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0))
== 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    int opt = 1;
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR,
&opt, sizeof(opt))) {
        perror("setsockopt");
        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) {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }
    if (listen(server_fd, 3) < 0) {
        perror("listen");
        exit(EXIT_FAILURE);
    }

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

    while (1) {
        if ((new_socket = accept(server_fd, (struct
sockaddr *) &address, (socklen_t * ) & addrlen)) < 0) {
            perror("accept");
            exit(EXIT_FAILURE);
        }

        printf("Client connected: %s:%d\n",
inet_ntoa(address.sin_addr), ntohs(address.sin_port));

        pthread_t thread;
        if (pthread_create(&thread, NULL,
handle_client, (void *) &new_socket) < 0) {
            perror("pthread_create");
            exit(EXIT_FAILURE);
        }

        pthread_detach(thread);
    }
```

```c
    return 0;
}

// Q2 client

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

#define MAX_EXPRESSION_LENGTH 100
#define SENTINEL_VALUE "exit"

char *resolve_hostname(const char *hostname) {
    struct hostent *host_entry;
    struct in_addr **addr_list;
    char *ip = NULL;

    if ((host_entry = gethostbyname(hostname)) == NULL) {
        perror("gethostbyname error!\n");
        return NULL;
    }

    addr_list = (struct in_addr **) host_entry->h_addr_list;
    if (addr_list[0] != NULL) {
        ip = strdup(inet_ntoa(*addr_list[0]));
    }

    return ip;
}

int main(int argc, char *argv[]) {
    if (argc != 3) {
        printf("Usage: %s <hostname> <port>\n", argv[0]);
        return 1;
    }

    const char *hostname = argv[1];
    const char *port = argv[2];
    char expression[MAX_EXPRESSION_LENGTH];
    char answer[MAX_EXPRESSION_LENGTH];
    char *server_ip;

    server_ip = resolve_hostname(hostname);
    if (server_ip == NULL) {
        fprintf(stderr, "Error: Unable to resolve hostname!\n");
        return 1;
    }

    printf("Server IP address: %s\n", server_ip);
    printf("Server Port: %s\n", port);

    int sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock < 0) {
        perror("socket error!\n");
        return 1;
    }

    struct sockaddr_in server_addr;
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(atoi(port));
    server_addr.sin_addr.s_addr = inet_addr(server_ip);

    if (connect(sock, (struct sockaddr *) &server_addr, sizeof(server_addr)) < 0) {
        perror("connect error!\n");
        return 1;
    }

    while (1) {
        printf("Enter an arithmetic expression to calculate (or '%s' to exit): ", SENTINEL_VALUE);
        fgets(expression, MAX_EXPRESSION_LENGTH, stdin);
        expression[strcspn(expression, "\n")] = '\0';

        if (strcmp(expression, SENTINEL_VALUE) == 0) {
            break;
        }

        send(sock, expression, strlen(expression), 0);
        int bytes_received = recv(sock, answer,
MAX_EXPRESSION_LENGTH, 0);
        if (bytes_received < 0) {
            perror("recv");
            break;
        }

        answer[bytes_received] = '\0';

        printf("Answer: %s\n", answer);
    }

    close(sock);
    free(server_ip);

    return 0;
}
```



```
…/net/2024-04-24/assgn11

(base) → ~/desktop/cse/A3300/sem6/net/2024-04-24/assgn11 $ clang q2srvr.c -o q2s
(base) → ~/desktop/cse/A3300/sem6/net/2024-04-24/assgn11 $ ./q2s localhost 50000
Server listening on port 50000
Client connected: 127.0.0.1:51861
Client connected: 127.0.0.1:51862
Client connected: 127.0.0.1:51863
^C
(base) → ~/desktop/cse/A3300/sem6/net/2024-04-24/assgn11 $
```

```
(base) → ~/desktop/cse/A3300/sem6/net/2024-04-24/as
sgn11 $ clang q2clnt.c -o q2c
(base) → ~/desktop/cse/A3300/sem6/net/2024-04-24/as
sgn11 $ ./q2c localhost 50000
Server IP address: 127.0.0.1
Server Port: 50000
Enter an arithmetic expression to calculate (or 'exi
t' to exit): 5+3
Answer: 8.00
Enter an arithmetic expression to calculate (or 'exi
t' to exit): exit
(base) → ~/desktop/cse/A3300/sem6/net/2024-04-24/as
sgn11 $
```

```
(base) → ~/desktop/cse/A3300/sem6/net/2024-04-24/as
sgn11 $ clang q2clnt.c -o q2c
(base) → ~/desktop/cse/A3300/sem6/net/2024-04-24/as
sgn11 $ ./q2c localhost 50000
Server IP address: 127.0.0.1
Server Port: 50000
Enter an arithmetic expression to calculate (or 'exi
t' to exit): 8/2
Answer: 4.00
Enter an arithmetic expression to calculate (or 'exi
t' to exit): exit
(base) → ~/desktop/cse/A3300/sem6/net/2024-04-24/as
sgn11 $
```

```
(base) → ~/desktop/cse/A3300/sem6/net/2024-04-24/as
sgn11 $ clang q2clnt.c -o q2c
(base) → ~/desktop/cse/A3300/sem6/net/2024-04-24/as
sgn11 $ ./q2c localhost 50000
Server IP address: 127.0.0.1
Server Port: 50000
Enter an arithmetic expression to calculate (or 'exi
t' to exit): 6-9
Answer: -3.00
Enter an arithmetic expression to calculate (or 'exi
t' to exit): exit
(base) → ~/desktop/cse/A3300/sem6/net/2024-04-24/as
sgn11 $
```