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

#define PORT "4321"
#define SIZE 512

int main(int argc, char **argv) {
        int sock, i, rc;
        char buffer[SIZE];

        struct sockaddr address;
        socklen_t addrLength = sizeof(address);
        struct addrinfo hints;
        struct addrinfo *addr;

        memset(&hints, 0, sizeof(hints));

        hints.ai_socktype = SOCK_DGRAM;
        hints.ai_flags = AI_PASSIVE | AI_ADDRCONFIG;
        if((rc = getaddrinfo(NULL, PORT, &hints, &addr))) {
                printf("host name lookup failed: %s\n", gai_strerror(rc));
                exit(1);
        }

        sock = socket(addr->ai_family, addr->ai_socktype, addr->ai_protocol);
        if(sock < 0) {
                printf("Can't create socket\n");
                exit(1);
        }

        i = 1;
        setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, &i, sizeof(i));

        rc = bind(sock, addr->ai_addr, addr->ai_addrlen);
        if(rc < 0) {
                printf("Can't bind socket\n");
                exit(1);
        }

        freeaddrinfo(addr);

        while(1) {
                int first = 0;
                int second = 0;

                rc = recvfrom(sock, buffer, SIZE, 0, (struct sockaddr*) &address, &addrLength);
                sscanf(buffer, "%d", &first);
                rc = recvfrom(sock, buffer, SIZE, 0, (struct sockaddr*) &address, &addrLength);
                sscanf(buffer, "%d", &second);

                printf("%d %d", second, first);
                second += first;
                printf("%d", second);

                sprintf(buffer, "%d", second);
                sendto(sock, buffer, rc, 0, (const struct sockaddr*) &address, addrLength);
        }

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

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

#define PORT 4321
#define SIZE 512

int main(int argc, char **argv) {
        struct addrinfo hints;
        struct addrinfo *addr;
        struct sockaddr_in *addrinfo;

        int rc, sock;
        char buffer[SIZE];
        char* ret;

        memset(&hints, 0, sizeof(hints));

        hints.ai_socktype = SOCK_DGRAM;
        hints.ai_flags = AI_ADDRCONFIG;

        rc = getaddrinfo("localhost", NULL, &hints, &addr);
        if(rc != 0) {
                printf("Host name lookup failed: %s\n", gai_strerror(rc));
                exit(1);
        }

        addrinfo = (struct sockaddr_in *) addr->ai_addr;
        sock = socket(addrinfo->sin_family, addr->ai_socktype, addr->ai_protocol);
        if(sock < 0) {
                printf("Can't create socket\n");
                exit(1);
        }

        addrinfo->sin_port = htons(PORT);
        freeaddrinfo(addr);

        while(1) {
                //get two numbers for sum
                for (int i = 0; i < 2; i++) {
                        ret = fgets(buffer, SIZE, stdin);
                        if(ret == NULL) break;
                        sendto(sock, buffer, strlen(buffer), 0, (const struct sockaddr*)
addrinfo, addr->ai_addrlen);
                }

                recvfrom(sock, buffer, SIZE, 0, NULL, NULL);
                printf("sum = %s\n", buffer);
        }

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

Lab 8

```
.PHONY: all clean
CFLAGS = -Wall -g

all: server client

server:
    cc -o server server.c

client:
    cc -o client client.c

clean:
    rm client server
```

Output

```
martin@LAPTOP-U1043R56:/mnt/c/Users/larry/Documents/versusCode/systems/lab/8$ ./server
^C
martin@LAPTOP-U1043R56:/mnt/c/Users/larry/Documents/versusCode/systems/lab/8$ ./client
2
2
sum = 4
^C
```