ASCII +4 encryption & decryption

```
//Client
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <string.h>
#define PORTNO 9999

int main(void)
{
    char buf[256];:
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    printf("Socket created...\n");

    struct sockaddr_in seraddr;
    seraddr.sin_family = AF_INET;
    seraddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    seraddr.sin_port = htons(PORTNO);
:
    int result = connect(sockfd, (struct
sockaddr*)&seraddr, sizeof(seraddr));
    if(result==-1)
    {
        printf("Connection error...\n");
        close(sockfd);
        exit(1);
    }
    printf("Data to send to server: ");
    scanf("%s", buf);

    for(int i = 0; i < 256; i++)
    {
        if(buf[i] == 0)
            break;
        else
            buf[i] = buf[i] + 4;
    }
    printf("Sending encrypted data to server...\n");
    write(sockfd, buf, sizeof(buf));
    printf("Closing socket and exiting...\n");
    close(sockfd);
    return 0;
}
```

Remove duplicates until stop

```
//client
//same headers as prev

#define PORTNO 9999

int main(void)
{
    char buf[256];

    //Create socket:
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    printf("Socket created...\n");

    //Set server address:
    struct sockaddr_in seraddr;
    seraddr.sin_family = AF_INET;
    seraddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    seraddr.sin_port = htons(PORTNO);

    //Send connection request to server:
    int result = connect(sockfd, (struct
sockaddr*)&seraddr, sizeof(seraddr));
    if(result==-1)
    {
        printf("Connection error...\n");
        close(sockfd);
        exit(1);
    }

    //Until stop:
    while(1)
    {
        //Get user sentence to send:
        printf("Enter sentence to send to server: ");
        gets(buf);
        puts(buf);
        printf("Sending sentence to server...\n");
        write(sockfd, buf, sizeof(buf));
        if(strcmp(buf, "Stop")==0)
        {
            printf("Closing and stopping...\n");
            close(sockfd);
            exit(0);
        }
        printf("Waiting for server response...\n");
        read(sockfd, buf, sizeof(buf));
        printf("Response from server: %s\n", buf);

    }
    return 0;
}
```

TCP Concurrent sort + pid

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/types.h>
#define PORTNO 9999
int main(void) {
    int buf[256];
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    printf("Socket created...\n");
    struct sockaddr_in seraddr;
    seraddr.sin_family = AF_INET;
    seraddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    seraddr.sin_port = htons(PORTNO);
    int result = connect(sockfd, (struct
sockaddr*)&seraddr, sizeof(seraddr));
    if(result == -1) {
        printf("Connection error...\n");
        close(sockfd);
        exit(1);
    }
    printf("Enter number of elements: ");
    int n;
    scanf("%d", &n);
    printf("Sending size to server...\n");
    buf[0] = n;
    write(sockfd, buf, sizeof(buf));

    for(int i = 0; i < n; i++) {
        printf("Entry #%d: ", i + 1);
        scanf("%d", &buf[i]);
    }
    printf("Sending array to server...\n");
    write(sockfd, buf, sizeof(buf));
    printf("Waiting for server to return sorted array...\n");
    read(sockfd, buf, sizeof(buf));
    printf("SORTED ARRAY RECEIVED:\n");

    for(int i = 0; i < n; i++) {
        printf("%d\t", buf[i]);
    }
    printf("\n");
    printf("PID = %d\n", buf[n]);

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

TCP concurrent remote math server
client

```
//same headers
#define PORTNO 9999

int main(void) {
    int buf[256];
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    printf("Socket created...\n");

    struct sockaddr_in seraddr;
    seraddr.sin_family = AF_INET;
    seraddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    seraddr.sin_port = htons(PORTNO);

    int result = connect(sockfd, (struct
sockaddr*)&seraddr, sizeof(seraddr));
    if(result == -1) {
        printf("Connection error...\n");
        close(sockfd);
        exit(1);
    }

    int a, b;
    char op;
    printf("Op 1: ");
    scanf("%d", &a);
    printf("Op 2: ");
    scanf("%d", &b);
    printf("Operator: ");
    scanf(" %c", &op);
    buf[0] = a;
    buf[1] = b;
    buf[2] = op;

    printf("Sending expression %d %c %d to server...\n",
a, op, b);
    write(sockfd, buf, sizeof(buf));

    int res;
    printf("Waiting for response from server...\n");
    read(sockfd, buf, sizeof(buf));
    res = buf[0];
    printf("Result = %d\n", res);

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

TCP Daytime
//client

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

#define PORTNO 9999

int main(void) {
    char buf[256];
    int buf2[256];
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    printf("Socket created!\n");

    struct sockaddr_in seraddr;
    seraddr.sin_family = AF_INET;
    seraddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    seraddr.sin_port = htons(PORTNO);

    int result = connect(sockfd, (struct
sockaddr*)&seraddr, sizeof(seraddr));
    if(result == -1) {
        printf("Connection error...\n");
        close(sockfd);
        exit(1);
    }

    printf("Waiting for server to provide time...\n");
    read(sockfd, buf, sizeof(buf));
    printf("Current date-time: %s\n", buf);
    read(sockfd, buf2, sizeof(buf2));
    printf("PID = %d\n", buf2[0]);

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

UDP matrix rows-SERVER

```
#define PORTNO 9999

int main(void)
{
    int buf[256];
    int matrix[256][256];
    int sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    printf("Socket created...\n");
    struct sockaddr_in seraddr, cliaddr;
    int clilen = sizeof(cliaddr);
    seraddr.sin_family = AF_INET;
    seraddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    seraddr.sin_port = htons(PORTNO);
    bind(sockfd, (struct sockaddr*)&seraddr,
sizeof(seraddr));
    printf("Socket binded!\n");
    int m, n;
    recvfrom(sockfd, buf, sizeof(buf), 0, (struct
sockaddr*)&cliaddr, &clilen);
    m = buf[0];
    n = buf[1];
    printf("Dimensions received: %d x %d\n", m, n);
    for(int i = 0; i < m; i++)
    {
        recvfrom(sockfd, buf, sizeof(buf), 0, (struct
sockaddr*)&cliaddr, &clilen);
        printf("Received row#%d: ", i + 1);
        for(int j = 0; j < n; j++)
        {
            printf("%d\t", buf[j]);
            matrix[i][j] = buf[j];
        }
        printf("\n");
    }
    printf("Sending matrix to client...\n");
    for(int i = 0; i < m; i++)
    {
        for(int j = 0; j < n; j++)
        {
            buf[i * n + j] = matrix[i][j];
        }
    }
    sendto(sockfd, buf, sizeof(buf), 0, (struct
sockaddr*)&cliaddr, sizeof(cliaddr));

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

TCP DAYTIME- Server

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

#define PORTNO 9999

int main(void) {
    char buf[256];
    int buf2[256];
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    printf("Socket created...\n");

    struct sockaddr_in seraddr, cliaddr;
    int clilen = sizeof(cliaddr);
    seraddr.sin_family = AF_INET;
    seraddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    seraddr.sin_port = htons(PORTNO);
    bind(sockfd, (struct sockaddr*)&seraddr,
sizeof(seraddr));
    printf("Server address binded to port...\n");

    listen(sockfd, 5);
    while(1) {
        int newsockfd = accept(sockfd, (struct
sockaddr*)&cliaddr, &clilen);
        int pid = fork();
        if(pid == 0) {
            time_t t;
            time(&t);
            printf("Current time: %s\n", ctime(&t));
            strcpy(buf, ctime(&t));
            printf("Sending to client...\n");
            write(newsockfd, buf, sizeof(buf));
            buf2[0] = getpid();
            write(newsockfd, buf2, sizeof(buf2));
            close(newsockfd);
            exit(0);
        } else {
            close(newsockfd);
        }
    }
    return 0;
}
```

UDP MATRIX ROWS-CLIENT

```
#define PORTNO 9999
int main(void)
{
    int buf[256];
    int sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    struct sockaddr_in seraddr;
    seraddr.sin_family = AF_INET;
    seraddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    seraddr.sin_port = htons(PORTNO);

    int m, n;
    printf("Enter dimensions of matrix: ");
    scanf("%d %d", &m, &n);
    buf[0] = m;
    buf[1] = n;
    sendto(sockfd, buf, sizeof(buf), 0, (struct
sockaddr*)&seraddr, sizeof(seraddr));

    for(int i = 0; i < m; i++)
    {
        printf("Enter data for Row#%d: ", i + 1);
        for(int j = 0; j < n; j++)
        {
            scanf("%d", &buf[j]);
        }
        sendto(sockfd, buf, sizeof(buf), 0, (struct
sockaddr*)&seraddr, sizeof(seraddr));
    }

    printf("Waiting for server matrix...\n");
    int serlen = sizeof(seraddr);
    recvfrom(sockfd, buf, sizeof(buf), 0, (struct
sockaddr*)&seraddr, &serlen);

    printf("MATRIX:\n");
    for(int i = 0; i < m; i++)
    {
        for(int j = 0; j < n; j++)
        {
            printf("%d\t", buf[i * n + j]);
        }
        printf("\n");
    }

    close(sockfd);
    return 0;
```

TCP CONCURRENT SORT+PID Server

```
#define PORTNO 9999
void sort(int arr[], int n) {
    for(int i = 0; i < n - 1; i++) {
        for(int j = 0; j < n - 1 - i; j++) {
            if(arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
}       }       }       }
int main(void) {
    int buf[256];
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    printf("Socket created...\n");
    struct sockaddr_in seraddr, cliaddr;
    int clilen = sizeof(cliaddr);
    seraddr.sin_family = AF_INET;
    seraddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    seraddr.sin_port = htons(PORTNO);
    bind(sockfd, (struct sockaddr*)&seraddr,
sizeof(seraddr));
    printf("Server address binded to socket...\n");
    listen(sockfd, 5);
    while(1) {
        int newsockfd = accept(sockfd, (struct
sockaddr*)&cliaddr, &clilen);
        int pid = fork();
        if(pid == 0) {
            int n;
            printf("Connection accepted!\n");
            read(newsockfd, buf, sizeof(buf));
            n = buf[0];
            printf("Size of array: %d\n", n);
            read(newsockfd, buf, sizeof(buf));
            sort(buf, n);
            printf("Sorted array:\n");
            for(int i = 0; i < n; i++) {
                printf("%d\t", buf[i]);
            }
            printf("\n");
            buf[n] = getpid();
            printf("Sending to client...\n");
            write(newsockfd, buf, sizeof(buf));
            printf("Closing and exiting...\n");
            close(newsockfd);
            exit(0);
        } else {
            close(newsockfd);
        }
    }
    return 0;
}
```

TCP CONCURRENT MATH - SERVER

```
#define PORTNO 9999
int main(void) {
    int buf[256];
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    printf("Socket created!\n");

    struct sockaddr_in seraddr, cliaddr;
    int clilen = sizeof(cliaddr);
    seraddr.sin_family = AF_INET;
    seraddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    seraddr.sin_port = htons(PORTNO);
    bind(sockfd, (struct sockaddr*)&seraddr,
sizeof(seraddr));
    printf("Server address binded to socket...\n");
    listen(sockfd, 5);
    while(1) {
        int newsockfd = accept(sockfd, (struct
sockaddr*)&cliaddr, &clilen);
        int pid = fork();
        if(pid == 0) {
            printf("Connection accepted!\n");
            int a, b;
            char op;
            read(newsockfd, buf, sizeof(buf));
            a = buf[0];
            b = buf[1];
            op = buf[2];
            printf("Received arithmetic expression: %d %c
%d\n", a, op, b);
            int res;
            switch(op) {
                case '+': res = a + b; break;
                case '-': res = a - b; break;
                case '*': res = a * b; break;
                case '/': res = a / b; break;
            }
            printf("Result = %d\n", res);
            printf("Sending to client...\n");
            buf[0] = res;
            write(newsockfd, buf, sizeof(buf));
            close(newsockfd);
            exit(0);
        } else {
            close(newsockfd);
        }
    }
    return 0;
}
```

ASCII 4+4 ENCRYPTION DECRYPTION

```
//SERVER
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/types.h>
#define PORTNO 9999

int main(void) {
    char buf[256];
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    printf("Socket created...\n");
    struct sockaddr_in seraddr, cliaddr;
    int clilen = sizeof(cliaddr);
    seraddr.sin_family = AF_INET;
    seraddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    seraddr.sin_port = htons(PORTNO);
    bind(sockfd, (struct sockaddr*)&seraddr,
sizeof(seraddr));
    printf("Socket binded to server address...\n");
    listen(sockfd, 5);
    int newsockfd = accept(sockfd, (struct
sockaddr*)&cliaddr, &clilen);
    printf("Connection accepted!\n");
    read(newsockfd, buf, sizeof(buf));
    printf("Encrypted message received from client: %s\n",
buf);
    for(int i = 0; i < 256; i++) {
        if(buf[i] == 0) {
            break;
        } else {
            buf[i] = buf[i] - 4;
    }  }
    printf("Decrypted message: %s\n", buf);
    printf("Closing socket and exiting...\n");
    close(newsockfd);
    close(sockfd);
    return 0;
}
```

REMOVE DUPLICATES UNTIL STOP - SERVER

```
#define PORTNO 9999
int main(void) {
    char buf[256];
    char words[256][256];
    char unique_words[256][256];
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    printf("Socket created...\n");
    struct sockaddr_in seraddr, cliaddr;
    int clilen = sizeof(cliaddr);
    seraddr.sin_family = AF_INET;
    seraddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    seraddr.sin_port = htons(PORTNO);
    bind(sockfd, (struct sockaddr*)&seraddr,
sizeof(seraddr));
    printf("Server address binded to socket...\n");
    listen(sockfd, 5);
    int newsockfd = accept(sockfd, (struct
sockaddr*)&cliaddr, &clilen);
    printf("Connection accepted...\n");
    while(1) {
        read(newsockfd, buf, sizeof(buf));
        printf("Message sent by client: %s\n", buf);
        if(strcmp(buf, "Stop") == 0) {
            printf("Closing and exiting...\n");
            close(newsockfd);
            close(sockfd);
            exit(0);
        }
        int curr_word = 0;
        char* token = strtok(buf, " ");
        while(token != NULL) {
            strcpy(words[curr_word], token);
            curr_word++;
            token = strtok(NULL, " ");
        }
        int total_count = curr_word;
        curr_word = 0;
        int unique_count = 0;
        for(int i = 0; i < total_count; i++) {
            int unique = 1;
            for(int j = 0; j < unique_count; j++) {
                if(strcmp(unique_words[j], words[i]) == 0) {
                    unique = 0;
                    break;
                }  }
            if(unique) {
                strcpy(unique_words[curr_word], words[i]);
                curr_word++;
                unique_count++;
            }  }
        memset(buf, 0, sizeof(buf));
        for(int i = 0; i < unique_count; i++) {
            strcat(buf, unique_words[i]);
            if(i < unique_count - 1) {
                strcat(buf, " ");
            }  }
        printf("Sending message to client...\n");
        write(newsockfd, buf, sizeof(buf));
    }

    return 0;}
```