```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#define WAIT_T 16
#define MAX 256

//SERVER

void append(char *str, char c, int idx) {
  int len = strlen(str);
  str[idx] = c;
}

char* nextASCII(char* str) {
  int value;
    int len = strlen(str);

  for(int i = 0; i < len; i++) {
    value = (int) str[i];
    value = value + 1;

    // append next character to string
    append(str, (char)value, i);
  }
    str[len + 1] = '\0';
  return str;
}

int main(int argc, char *argv[]) {
    // port to start the server on
    int SERVER_PORT = 8877;

    // build address data structure
    struct sockaddr_in server_address;
    memset(&server_address, 0, sizeof(server_address));
    server_address.sin_family = AF_INET;
    server_address.sin_port = htons(SERVER_PORT);
    server_address.sin_addr.s_addr = htonl(INADDR_ANY);

    // create open socket
    int listen_sock;
    if ((listen_sock = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
        printf("could not create listen socket\n");
        return 1;
    }

    // bind socket to listen to connection
```

```
  if ((bind(listen_sock, (struct sockaddr *)&server_address,
            sizeof(server_address))) < 0) {
      printf("could not bind socket\n");
      return 1;
  }

  int wait_size = 16;
// listen to socket, print error if could not open socket
  if (listen(listen_sock, wait_size) < 0) {
      printf("could not open socket for listening\n");
      return 1;
  }

  // socket address for client
  struct sockaddr_in client_address;
  socklen_t client_address_len = 0;

  while (1) {
      // open socket to accept data
      int sock;
      if ((sock = accept(listen_sock, (struct sockaddr *)&client_address,
                    &client_address_len)) < 0) {
          printf("could not open a socket to accept data\n");
          return 1;
      }

      int i = 0;
      int len = 0, max = 100;
      char buff[max];
      char *pbuff = buff;

      printf("client connected with ip address: %s\n",
             inet_ntoa(client_address.sin_addr));

      // run as long as there is a connection to client
      while ((i = recv(sock, pbuff, max, 0)) > 0) {
          pbuff += i;
          max -= i;
          len += i;
    buff[len] = '\0';

    char* str = malloc(strlen(buff));
          printf("received: '%s'\n", buff);

    // find next ascii character of string
    str = nextASCII(buff);
          send(sock, buff, len, 0);
    break;
      }
      shutdown(sock,0);
  }
```

```
    shutdown(listen_sock,0);
    return 0;
}
```