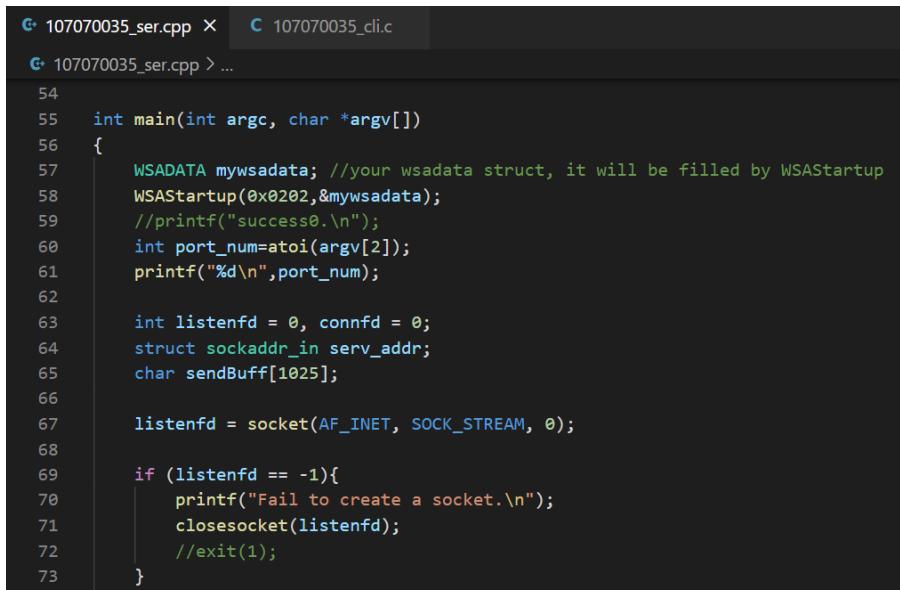


1. Details of your implementation, including server-side and client-side.

【server端】



```
54 int main(int argc, char *argv[])
55 {
56     WSADATA mywsadata; //your wsadata struct, it will be filled by WSASStartup
57     WSAStartup(0x0202,&mywsadata);
58     //printf("success0.\n");
59     int port_num=atoi(argv[2]);
60     printf("%d\n",port_num);
61
62     int listenfd = 0, connfd = 0;
63     struct sockaddr_in serv_addr;
64     char sendBuff[1025];
65
66     listenfd = socket(AF_INET, SOCK_STREAM, 0);
67
68     if (listenfd == -1){
69         printf("Fail to create a socket.\n");
70         closesocket(listenfd);
71         //exit(1);
72     }
73 }
```

- 根據tutorial, 在使用winsock前需先呼叫WSADATA、WSAStartup
- 60行是將輸入的port轉型太, 之後才能使用
- 63~65創建等一下需要的參數
- 67行建立socket, 使用int socket(int domain, int type, int protocol)的protocol。domain是定義了socket的溝通領域, 我使用的是AF_INET, 使用IPv4讓兩台主機透過網路來傳輸;type定義了socket的傳輸方式, SOCK_STREAM代表是提供一個序列化的連接導向位元流, 對應了TCP;最後protocol是設定socket的標準, 這邊設為0讓kernel自行選擇。而listenfd存取了socket()return value, 也就是socket file descriptor。
- 若socket創建失敗則會回傳-1, 69~73就是在檢查是否失敗。

```
107070035_ser.cpp X 107070035_cli.c
107070035_ser.cpp > ...
75     memset(&serv_addr, '0', sizeof(serv_addr));
76     serv_addr.sin_family = AF_INET;
77     serv_addr.sin_addr.s_addr = INADDR_ANY;
78     serv_addr.sin_port = htons(port_num);
79     //printf("success2.\n");
80     if (bind(listenfd, (struct sockaddr*)&serv_addr, sizeof(serv_addr)) == -1){
81         printf("Fail to bind a socket.\n");
82         closesocket(listenfd);
83         exit(1);
84     }
85     listen(listenfd, 10);
86     //printf("success3.\n");
87
88     while(1)
89     {
90         connfd = accept(listenfd, (struct sockaddr*)NULL, NULL);
91         if (connfd == -1){
92             printf("error at connection.\n");
93             exit(1);
94         }
95         //printf("new round\n");
96         //printf("accept success\n");
97         /*char word='S';
98         send(connfd, &word,sizeof(word),0);*/
99         game_loop(connfd);
100        printf("Next Round\n");
101        close(connfd);
102    }
103 }
```

- server負責提供socket的所有訊息，因此需要設定。memset一個空間給serv_saddr，根據tutorial給的資料來設定此data structure。
- 76行，
- 77行的.sin_addr.s_addr = INADDR_ANY，裡面的INADDR_ANY讓kernel決定local IP。
- 78行，使用的htons是將local端的字節序轉換成網路使用的。
- 80~84行在做int bind(int sockfd, struct sockaddr* addr, int addrlen);，目的是把是把設定的address綁在Socket身上，sockfd是socket的描述符，第二個參數addr就是把剛剛創建的serv_addr船進去，最後是addrlen就是描述addr的大小。
Return value是0/-1，0代表成功，-1代表失敗。
- 88~102是遊戲的實作，在開始玩遊戲之前需要先讓socket接收client，於是乎較int accept(int sockfd, struct sockaddr addr, socklen_t addrlen);這裡第一個參數依樣是該socket，第二個是socket的結構，用於儲存接收到的client端資訊，最後是addr的大小。這邊會return一個新的socket，也就是跟client溝通的socket file descriptor。
- 99行這邊我呼叫了game_loop，我自己寫的function。以下為gameloop實作

```

G: 107070035_ser.cpp > ...
12 void game_loop(int fd){
13     printf("=====\\n");
14     printf("Game Start\\n");
15     printf("=====\\n");
16
17
18     srand(time(NULL));
19     int random_number = (rand() % 99)+1;
20     //printf("%d",random_number);
21     char buff[256]={};
22     int converted_number;
23     char word='0';
24     word='S';
25     send(fd, &word,sizeof(word),0);

```

- 設定random，並random出一個數字，當作猜數字的答案。
- 24行的word是server會送給client的訊息，25行送出'S'代表遊戲Start。Client端會用recv來接收server的資訊。

```

G: 107070035_ser.cpp > ...
24     word='S';
25     send(fd, &word,sizeof(word),0);
26     while (1) {
27         sleep(1);
28         if (recv(fd,buff,sizeof(buff),0)==-1){
29             printf("error at recv.\\n");
30         }
31         //printf("%s ",buff);
32         converted_number = atoi(buff);
33         //printf("%d\\n",converted_number);
34         if (converted_number == random_number) {
35             word='=';
36             //printf("=\n");
37             printf("Correct\\n");
38             send(fd,&word,sizeof(word),0);
39             return;
40         }else if (converted_number > random_number) {
41             word='>';
42             //printf(">\\n");
43             send(fd,&word,sizeof(word),0);
44         }else if (converted_number < random_number) {
45             word='<';
46             //printf("<\\n");
47             send(fd,&word,sizeof(word),0);
48         }else {
49             printf("error at guess\\n");
50             exit(1);
51         }

```

- 26行開始是一輪遊戲，用while loop直到此輪遊戲結束。
- 28行是在接收client端的訊息，也就是client端會傳一個數字過來，這邊我是用字串形式接收。如果沒收到會return -1。32行是轉換client傳來的數字成int。
- 34-51是遊戲的比較，如果猜對會傳送'='給client，並return結束此run。
- 若client猜的數字較大，傳送'>'，while loop繼續；若client猜的數字較小，傳送'<'，while loop繼續。

【client端】

```
C 107070035_cli.c > main(int, char *[])
12
13     int main(int argc, char *argv[]){
14
15         WSADATA mywsadata; //your wsadata struct, it will be filled by WSAStartup
16         WSAStartup(0x0202,&mywsadata);
17
18         int sockfd=0;
19         char buff[256]={};
20         char receive='0';
21         //char port_num[]={};
22         int port_num=atoi(argv[3]);
```

- 根據tutorial, 在使用winsock前需先呼叫WSADATA、WSAStartup
- 設應sock的variable, 這邊依樣要將port_num轉形態

```
C 107070035_cli.c > main(int, char *[])
30     while(1){
31         sockfd=socket(AF_INET , SOCK_STREAM , 0);
32         //listenfd = socket(AF_INET, SOCK_STREAM, 0);
33         if (sockfd == -1){
34             printf("Fail to create a socket.\n");
35         }
36
37         struct sockaddr_in serv_addr;
38         memset(&serv_addr, '0', sizeof(serv_addr));
39         serv_addr.sin_family = AF_INET;
40         serv_addr.sin_port = htons(port_num);
41         serv_addr.sin_addr.s_addr = inet_addr(argv[2]);
42
43
44         int err= connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
45         if (err ==-1){
46             printf("connection error!\n");
47         }else{
48             printf("Connection Success.\n");
49         }
50
51         int game_start=1;
```

- 進入while loop, 會一直玩到client停止玩。31-41行也開始設定socket, 跟server基本上都一樣
- 44-51開始呼叫connect, 進行與server的連線。若是沒有連成功的話就會印出error

```
51         int game_start=1;
52
53
54         recv(sockfd,&receive,1,0);
55         //printf("%c",receive);
56         if(receive!='S'){
57             printf("%c",receive);
58             printf("Error at Start message.\n");
59             game_start=0;
60             exit(1);
61         }
```

- 54行開始此輪遊戲的設定, 要先收到server傳送的'S'才代表遊戲開始, 若是沒開始game_start設為0, 影響到等一下的while loop

```

64     int smallnumber=1,bignumber=100;
65     while(game_start){
66         printf("-----\n\n");
67         printf("Guess a number:\n");
68         scanf("%s",&buff);
69         //printf("%s",buff);
70         //converted_number=htonl(guessed_number);
71         send(sockfd,buff,sizeof(buff),0);
72         //int converted_number = htonl(guessed_number);
73         //send(sockfd, &converted_number, sizeof(uint32_t),0);
74
75         recv(sockfd,&receive,1,0);
76         if(receive=='='){
77             printf("Answer Correct!\n\n");
78             printf("=====*\n");
79             closesocket(sockfd);
80             //receive='S';
81             sleep(1);
82             break;
83         }else if(receive=='>'){
84             int tmp=atoi(buff);
85             if(tmp<=bignumber) bignumber=tmp;
86             printf("Lower than %d\nHigher than %d\n", bignumber,smallnumber);
87         }else if(receive=='<'){
88             int tmp=atoi(buff);
89             if(tmp>smallnumber) smallnumber=tmp;
90             printf("Lower than %d\nHigher than %d\n", bignumber,smallnumber);
91         }else{
92             printf("Error at receive.\n");
93             closesocket(sockfd);
94             exit(1);
95         }
96     }
97 }
98 WSACleanup();
99 }
```

- whileloop主要是實作一輪遊戲。設定好smallnumber、bignumber代表目前的低上限
- 68行請玩家輸入數字，並將數字send給server。75行收到server判斷出來的結果。
- 如果是'='就代表猜對，呼叫close socket將連線刪除並離開此run遊戲，進入下一輪。
- 如果是'>' or '<' 就繼續猜，並更新smallnum、bignum

2. The answers to the Wireshark observation.

- Show the packets used for TCP hand shaking.

| ip.addr == 127.0.0.1 && tcp.port==2002 | | | | | |
|--|-------------|----------|--------|--|--|
| | Destination | Protocol | Length | Info | |
| .1 | 127.0.0.1 | TCP | 56 | 61691 → 2002 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SAC... | |
| .1 | 127.0.0.1 | TCP | 56 | 2002 → 61691 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495... | |
| .1 | 127.0.0.1 | TCP | 44 | 61691 → 2002 [ACK] Seq=1 Ack=1 Win=2619648 Len=0 | |

這三個就是hand shacking的packets

- Show the server's IP and the client's IP.

```

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 61691, Dst Port: 2002, Seq: 0, Len: 0
```

server和client的IP可以手動輸入想要的IP 這邊我輸入127.0.0.1

- Show the server's port and the client's port.

```

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 61691, Dst Port: 2002, Seq: 0, Len: 0
```

server's port是2002, Client's port是61691

- Show the size of the packet transmitted by the client. (in bytes)

| Destination | Protocol | Length | Info |
|-------------|----------|--------|--------------|
| 127.0.0.1 | TCP | 44 | 61691 → 2002 |
| 127.0.0.1 | TCP | 300 | 61691 → 2002 |
| 127.0.0.1 | TCP | 44 | 2002 → 61691 |
| 127.0.0.1 | TCP | 45 | 2002 → 61691 |
| 127.0.0.1 | TCP | 44 | 61691 → 2002 |
| 127.0.0.1 | TCP | 300 | 61691 → 2002 |
| 127.0.0.1 | TCP | 44 | 2002 → 61691 |
| 127.0.0.1 | TCP | 45 | 2002 → 61691 |
| 127.0.0.1 | TCP | 44 | 61691 → 2002 |
| 127.0.0.1 | TCP | 300 | 61691 → 2002 |
| 127.0.0.1 | TCP | 44 | 2002 → 61691 |
| 127.0.0.1 | TCP | 45 | 2002 → 61691 |

當client送給server時，長度可能是44或300。

- How many routers does each of the transmitted packets goes through. (Hint: TTL)

```
Identification: 0x23c9 (9161)
> Flags: 0x40, Don't fragment
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 128
Protocol: TCP (6)
Header Checksum: 0x0000 [validation disabled]
```

因為TTL是128, window預設的TTL也是128, 代表沒經過router

3. Descriptions of difficulties you encountered and your solutions.

這個作業最難的點是我一開始不太清楚socket到底在哪個時候有連接，導致遊戲只能玩一run，之後想清楚後就知道while loop要怎麼改；另一個點是這個作業雖然tutorial寫得蠻清楚的，但還是要全盤理解才知道怎麼寫，所以花了一點時間去了解每個function的目的和意義，也因為要下載一些東西，需要很詳細的去看助教給的內容。最後demo發現自己把port num寫死了，很感謝demo的助教給我額外時間修code，順利完成這次作業。