PROGRAM CODE

server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <pthread.h>

#define PORT 8000
#define SIZE 100

typedef struct packet {
      int data;
      int type; // SEQ (0), ACK (1) or NACK(-1)
      int seq; // Sequence number
} packet;

void main() {
      int server_fd, client_fd;
      struct sockaddr_in address;
      int addrlen = sizeof(address);
      int arr[SIZE];

      for(int i = 0; i < SIZE; i++)
            arr[i] = -1;

      printf("Go-Back-N ARQ\nTCP Server\n");

      if((server_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
            printf("Socket creation failed!\n");
            exit(1);
      }

      address.sin_family = AF_INET;
      address.sin_addr.s_addr = INADDR_ANY;
      address.sin_port = htons(PORT);

      if(bind(server_fd, (struct sockaddr*) &address, addrlen) < 0) {
            printf("Socket binding failed!\n");
            exit(1);
      }

      if(listen(server_fd, 5) < 0) {
            printf("Listening failed!\n");
            exit(1);
      }

      if((client_fd = accept(server_fd, (struct sockaddr*) &address,
(socklen_t*) &addrlen)) < 0) {
            printf("Connection failed!\n");
            exit(1);
      } else {
            printf("Connected to client.\n");
      }

      packet p;
      int exp_seq = 0, flag = 0;

      while(1) {
            int status = recv(client_fd, &p, sizeof(packet), 0);

            if(status < 0) {
                  printf("Receive failed!\n");
            } else if (status == 0) {
                  printf("Receive completed.\nArray: ");

                  for(int i = 0; arr[i] != -1; i++) {
                        printf("%d ", arr[i]);
                  }

                  printf("\n");

                  break;
            } else {
                  if(p.seq > exp_seq) {
                        if(!flag) {
                              flag = 1;

                              p.type = -1;

                              p.seq = exp_seq;

                              if(send(client_fd, &p, sizeof(packet), 0) < 0) {
                                    printf("Send failed!\n");
                              } else {
                                    printf("Sent: NACK %d\n", p.seq);
                              }
                        }

                        continue;
                  } else {
                        flag = 0;

                        exp_seq = p.seq + 1;
                  }

                  p.type = 1;

                  printf("Received: %d (SEQ %d)\n", p.data, p.seq);
                  arr[p.seq] = p.data;

                  if(rand() % 10 != 6) {
                        if(send(client_fd, &p, sizeof(packet), 0) < 0) {
                              printf("Send failed!\n");
                        } else {
                              printf("Sent: ACK %d\n", p.seq);
                        }
                  } else {
                        printf("ACK %d lost\n", p.seq);
                  }
            }
      }

      close(server_fd);
      close(client_fd);
}
```

client.c

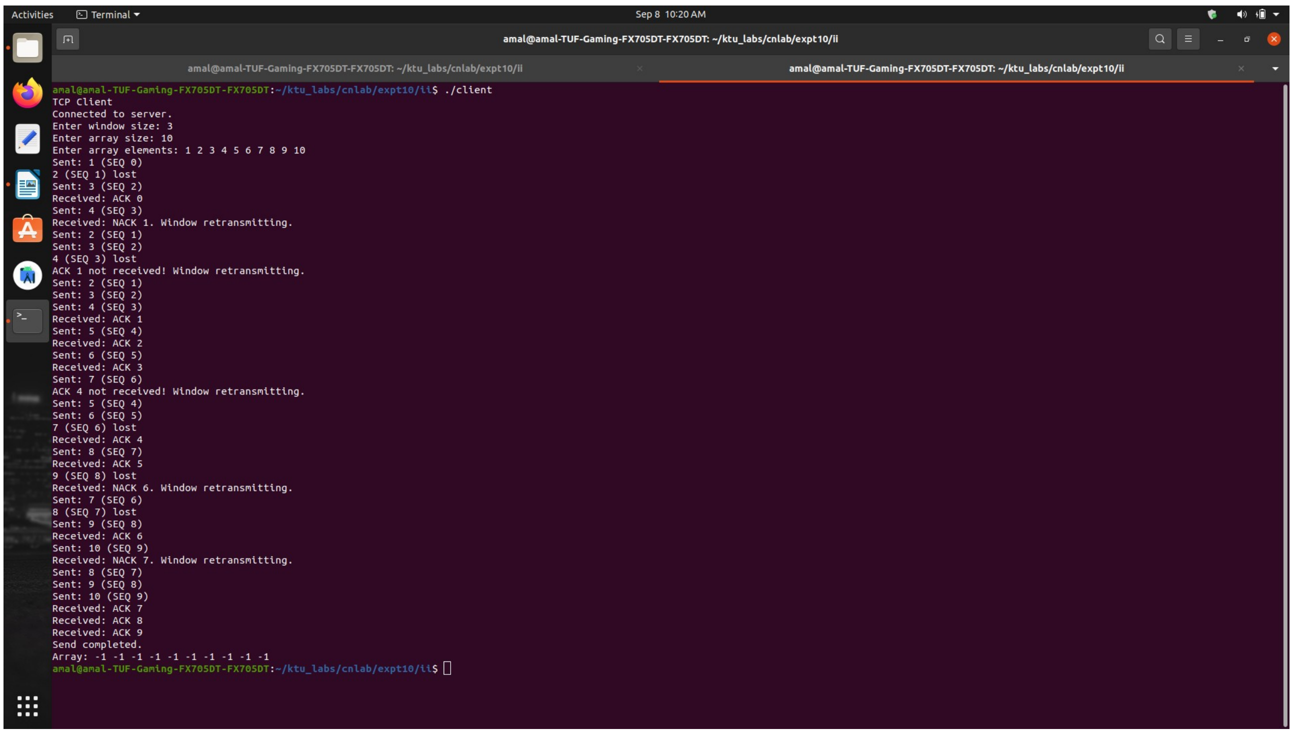
```
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <arpa/inet.h>
```

```c
#include <unistd.h>

#define PORT 8000

typedef struct packet {
      int data;
      int type; // SEQ (0), ACK (1) or NACK(-1)
      int seq; // Sequence number
} packet;

typedef struct window {
      int size;
      int start;
      int end;
} window;

typedef struct data {
      int* arr;
      int n;
      int client_fd;
      int exp_seq;
      packet* p;
      window* w;
} data;

void recvAck(data d);

void sendWindow(data d) {
      d.p->seq = d.w->start;

      for(int i = d.w->start; i <= d.w->end && i < d.n; i++) {
            d.p->type = 0;
            d.p->data = d.arr[i];

            if(rand() % 10 != 6) {
                  if(send(d.client_fd, d.p, sizeof(packet), 0) < 0) {
                        printf("Send failed!\n");
                  } else {
                        printf("Sent: %d (SEQ %d)\n", d.p->data, d.p->seq);
                  }
            } else {
                  printf("%d (SEQ %d) lost\n", d.p->data, d.p->seq);
            }

            d.p->seq = d.p->seq + 1;
      }

      recvAck(d);
}

void sendFrame(data d) {
      d.p->type = 0;
      d.p->data = d.arr[d.w->end];

      if(rand() % 10 != 6) {
            if(send(d.client_fd, d.p, sizeof(packet), 0) < 0) {
                  printf("Send failed!\n");
            } else {
                  printf("Sent: %d (SEQ %d)\n", d.p->data, d.p->seq);
            }
      } else {
            printf("%d (SEQ %d) lost\n", d.p->data, d.p->seq);
      }
```

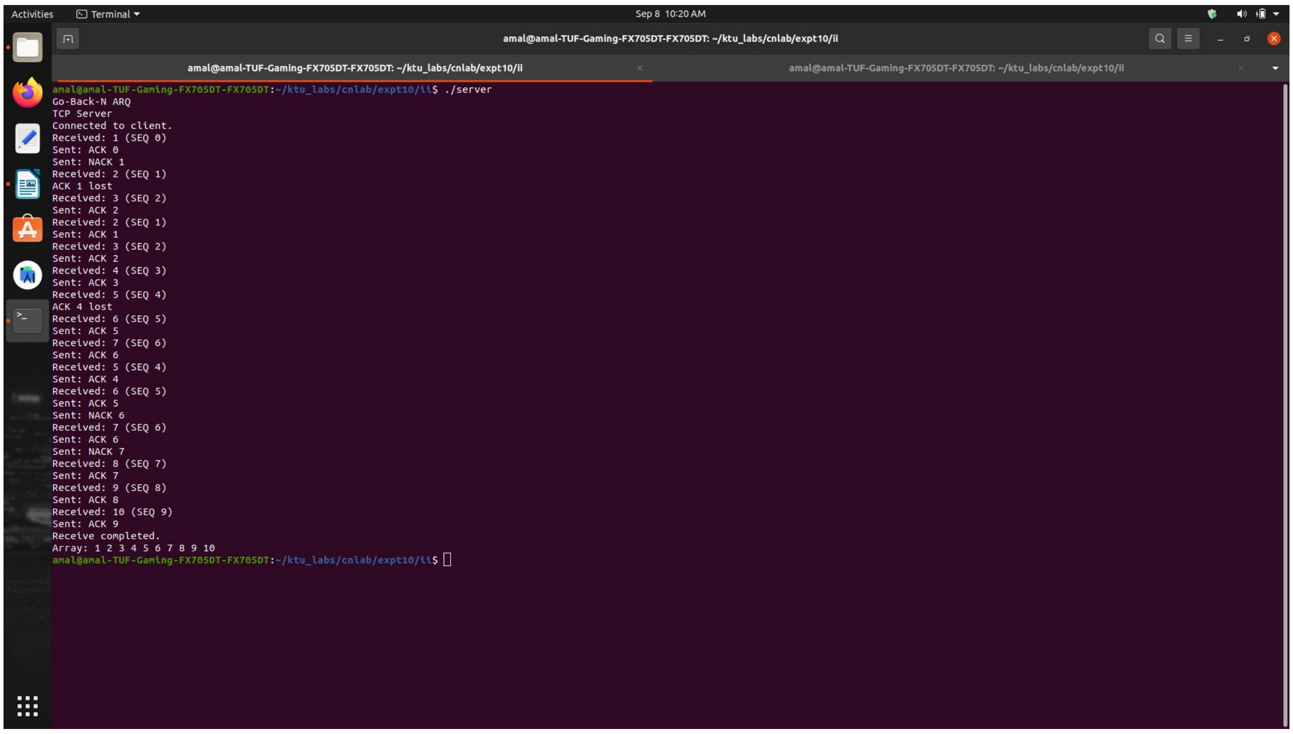
```
	d.p->seq = d.p->seq + 1;

	recvAck(d);
}

void recvAck(data d) {
	data d1;
	packet p;
	d1.p = &p;

	if(recv(d.client_fd, d1.p, sizeof(packet), 0) < 0) {
		printf("Time out! Window retransmitting.\n");
		sendWindow(d);
	} else {
		if(d1.p->seq > d.exp_seq) {
			printf("ACK %d not received! Window retransmitting.\n", 
d.exp_seq);

			while(recv(d.client_fd, d1.p, sizeof(packet), 0) > 0);

			sendWindow(d);

			return;
		}

		if(d1.p->type == 1) {
			printf("Received: ACK %d\n", d1.p->seq);

			d.arr[d1.p->seq] = -1;

			d.w->start++;

			if(d.w->start == d.n) {
				printf("Send completed.\nArray: ");

				for(int i = 0; i < d.n; i++) {
					printf("%d ", d.arr[i]);
				}

				printf("\n");

				close(d.client_fd);
				exit(0);
			}

			d.w->end++;

			d.exp_seq = d1.p->seq + 1;

			if(d.w->end < d.n)
				sendFrame(d);
			else
				recvAck(d);
		} else if(d1.p->type == -1) {
			printf("Received: NACK %d. Window retransmitting.\n", d1.p-
>seq);
			sendWindow(d);
		}
	}
}

void main() {
	int client_fd;
	struct sockaddr_in serv_addr;
```

```
      printf("TCP Client\n");

      client_fd = socket(AF_INET, SOCK_STREAM, 0);

      if(client_fd < 0) {
            printf("Socket creation failed!\n");
            exit(1);
      }

      serv_addr.sin_family = AF_INET;
      serv_addr.sin_addr.s_addr = INADDR_ANY;
      serv_addr.sin_port = htons(PORT);

      if(connect(client_fd, (struct sockaddr*) &serv_addr, sizeof(serv_addr)) <
0) {
            printf("Connection failed!\n");
            exit(1);
      } else {
            printf("Connected to server.\n");
      }

      struct timeval tv;
      tv.tv_sec = 1;
      tv.tv_usec = 0;
      setsockopt(client_fd, SOL_SOCKET, SO_RCVTIMEO, (const char*)&tv, sizeof
tv);

      int n;
      window w;

      printf("Enter window size: ");
      scanf("%d", &w.size);

      w.start = 0;
      w.end = w.size - 1;

      printf("Enter array size: ");
      scanf("%d", &n);

      int arr[n];

      printf("Enter array elements: ");
      for(int i = 0; i < n; i++) {
            scanf("%d", &arr[i]);
      }

      packet p;
      data d;
      d.client_fd = client_fd;
      d.p = &p;
      d.w = &w;
      d.n = n;
      d.arr = arr;
      d.exp_seq = 0;
      p.seq = 0;

      sendWindow(d);
}
```

OUTPUT


Activities
Terminal
Sep 8 10:20 AM
amal@amal-TUF-Gaming-FX705DT-FX705DT: ~/ktu_labs/cnlab/expt10/ii
amal@amal-TUF-Gaming-FX705DT-FX705DT: ~/ktu_labs/cnlab/expt10/ii
amal@amal-TUF-Gaming-FX705DT-FX705DT: ~/ktu_labs/cnlab/expt10/ii

```
amal@amal-TUF-Gaming-FX705DT-FX705DT:~/ktu_labs/cnlab/expt10/ii$ ./client
TCP Client
Connected to server.
Enter window size: 3
Enter array size: 10
Enter array elements: 1 2 3 4 5 6 7 8 9 10
Sent: 1 (SEQ 0)
2 (SEQ 1) lost
Sent: 3 (SEQ 2)
Received: ACK 0
Sent: 4 (SEQ 3)
Received: NACK 1. Window retransmitting.
Sent: 2 (SEQ 1)
Sent: 3 (SEQ 2)
4 (SEQ 3) lost
ACK 1 not received! Window retransmitting.
Sent: 2 (SEQ 1)
Sent: 3 (SEQ 2)
Sent: 4 (SEQ 3)
Received: ACK 1
Sent: 5 (SEQ 4)
Received: ACK 2
Sent: 6 (SEQ 5)
Received: ACK 3
Sent: 7 (SEQ 6)
ACK 4 not received! Window retransmitting.
Sent: 5 (SEQ 4)
Sent: 6 (SEQ 5)
7 (SEQ 6) lost
Received: ACK 4
Sent: 8 (SEQ 7)
Received: ACK 5
9 (SEQ 8) lost
Received: NACK 6. Window retransmitting.
Sent: 7 (SEQ 6)
8 (SEQ 7) lost
Sent: 9 (SEQ 8)
Received: ACK 6
Sent: 10 (SEQ 9)
Received: NACK 7. Window retransmitting.
Sent: 8 (SEQ 7)
Sent: 9 (SEQ 8)
Sent: 10 (SEQ 9)
Received: ACK 7
Received: ACK 8
Received: ACK 9
Send completed.
Array: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
amal@amal-TUF-Gaming-FX705DT-FX705DT:~/ktu_labs/cnlab/expt10/ii$
```


Activities
Terminal
Sep 8 10:20 AM
amal@amal-TUF-Gaming-FX705DT-FX705DT: ~/ktu_labs/cnlab/expt10/ii
amal@amal-TUF-Gaming-FX705DT-FX705DT: ~/ktu_labs/cnlab/expt10/ii
amal@amal-TUF-Gaming-FX705DT-FX705DT: ~/ktu_labs/cnlab/expt10/ii

```
amal@amal-TUF-Gaming-FX705DT-FX705DT:~/ktu_labs/cnlab/expt10/ii$ ./server
Go-Back-N ARQ
TCP Server
Connected to client.
Received: 1 (SEQ 0)
Sent: ACK 0
Sent: NACK 1
Received: 2 (SEQ 1)
ACK 1 lost
Received: 3 (SEQ 2)
Sent: ACK 2
Received: 2 (SEQ 1)
Sent: ACK 1
Received: 3 (SEQ 2)
Sent: ACK 2
Received: 4 (SEQ 3)
Sent: ACK 3
Received: 5 (SEQ 4)
ACK 4 lost
Received: 6 (SEQ 5)
Sent: ACK 5
Received: 7 (SEQ 6)
Sent: ACK 6
Received: 5 (SEQ 4)
Sent: ACK 4
Received: 6 (SEQ 5)
Sent: ACK 5
Sent: NACK 6
Received: 7 (SEQ 6)
Sent: ACK 6
Sent: NACK 7
Received: 8 (SEQ 7)
Sent: ACK 7
Received: 9 (SEQ 8)
Sent: ACK 8
Received: 10 (SEQ 9)
Sent: ACK 9
Receive completed.
Array: 1 2 3 4 5 6 7 8 9 10
amal@amal-TUF-Gaming-FX705DT-FX705DT:~/ktu_labs/cnlab/expt10/ii$
```