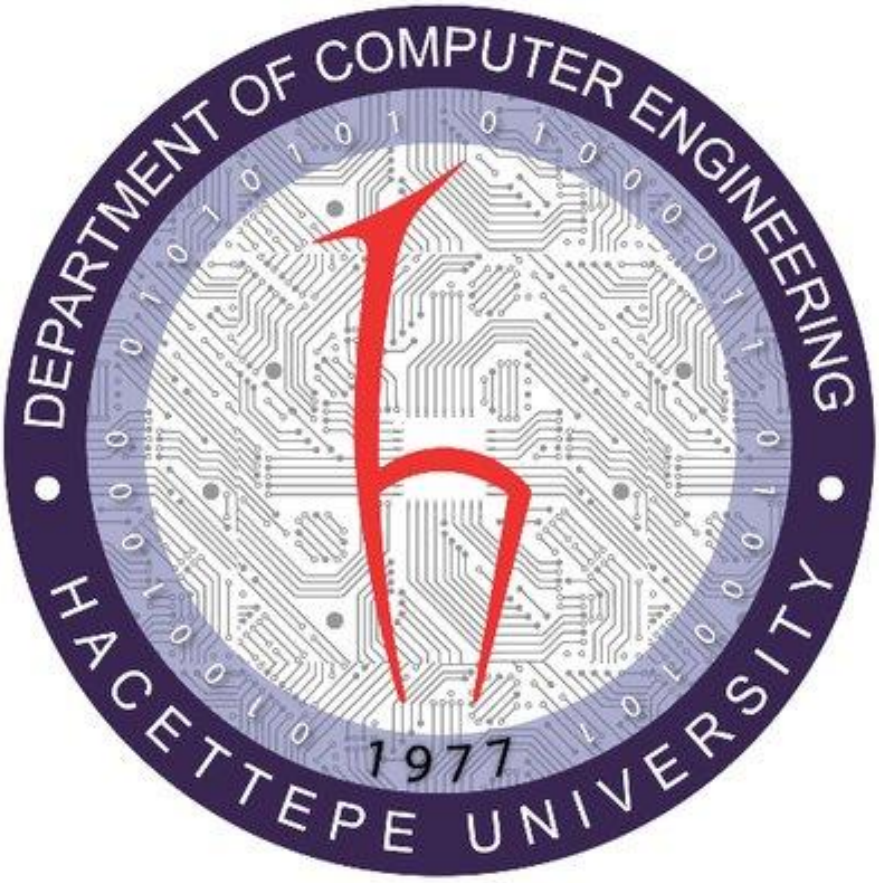
Hacettepe University
Department Of Computer Engineering



BBM203 Assignment2 Report

Name & Surname : *Tolga Furkan GÜLER*

Identity Number : *21692874*

Course : *BBM203 Software Laboratory*

Subject : Stack, Queue, and Dynamic Memory Allocation

PROBLEM DEFİNİTİON:

In this experiment, we designed and understand how a simple network connection works.We have 3 input files to use when doing this assignment.
First of all “Clients.dat”, it includes how many clients there are and their informations.The second one is “Routing.dat”,it includes each clients routing table.This tables show us the links between the clients.We can find which route we need to use thanks for these routing tables.The last one is “Command.dat”,it includes the commands which are we need to handle in this assignment.We may get command like Message sender client to receiver client and message, which is needed to send receiver client.After this command we need to find there is route or not between this clients, if there is we need to split this message and create the frames.Frames include 4 layer and each layer have some informations.Number of frames is equal the number of how many splitted message there are. After we create the frames we should send these frames to receiver in order.

Step by Step this Assignment:

- Read Clients and Routing file and fill the each clients informations.
- Read Commands file and take the “Message sender client #message#”command.
- Find route with a recursive function
- Split the message and create the frames as a stack abstarct data type.
- Add the frames to the queue
- Read the command file up to end of the file and handle each command using informations which are we have.

METHODS AND SOLUTİON:

These are my structs ı created these objects and use them in some funtions.

```
//Struct for clients
struct Clients {
    char  Id[MAX];
    char  Ip_adress[MAX];
    char  Mac_adress[MAX];
    char  routing_table[MAX][MAX];
    struct Queue* in;
    struct Queue* out;
};
//Struct for layers
struct Layer {
     char** info;
     struct Layer* next;
};
//Struct for stacks, top holds the top of the every stack
struct Stack{
     struct Layer* top;
     struct Stack* next;
};
//Struct for Queues
struct Queue{
     struct Stack* front;
     struct Stack* rear;
};
```

- This is my push function.I get the layer and stack type of variables and every push I ubdated the stack->top.Some layers have 2 info and some of them 3 therefore I take the info size as a parameter.

```
//This function pushes each layer to stack
struct Stack* push(struct Stack *stack,struct Layer *layer,int info_size){
```

- This my pop function.I decreased and updated stack->top every pop.

```
//This funtion pop the top of the stack
struct  Layer* pop(struct Stack *stack,struct Layer *layer)
```

- This my enqueue function.I get the stack and increased the rear(point to next) every enqueue.

```
//This function remove a stack from rear of queue
void enqueue(struct Stack* s,struct Queue* q){
```

- This my dequeue function.I increased the front(point to next) every dequeue.

```
//This function add stack to front od queue
void dequeue(struct Stack* s,struct Queue* q){
```

SOLUTİON:

I read the all files line by line up to end of the file with fgets function.After I get the lines I split the lines with strtok function.First of all I allocated memory for each clients, after that I read the Clients file line by line in a for loop and filled each clients informations with datas which are in Clients file. Subsequently I read the Routing file line by line and filled every clients routing table.After that I read the Command file and get the first line.in a while loop ı get the each line of the file and thanks for some “if-else” statement I handled them sparately.First command should be “Message sender receiver #message#” in this commands if statement I use sender and receiver clients informations which are ı filled before.Respectively I found route thanks for a recursive function.This function call itself until the receiver id equals the sender id, every call I updated sender id.After that I splitted message and crate frames with push function which is above.After creating frames I add each frame into queue with enqueue function which is above. After this command I have frames and queues so I can use them in other command,so I printed

needed output to console in other commands if-else statement.End of the assignment I close the files and free the memory which are allocated before.

Tolga Furkan GÜLER

21692874