HACETTEPE UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING BBM104

Name : Mehmet Taha

Surname : Usta

Number : 21527472

E~mail : b21527472@cs.hacettepe.edu.tr

Subject : Stack,Queue,Dynamic Memory Allocation

Programming Language : C

1.Aim

Build server and customer structure using stack structure,queue structure and dynamic memory allocation concept. Understand how stacks and queues work by sending commands to server and client

2. Problem solving

Firstly circular queues and stack are defined in struct. Then stack and circular queue methods were created. when we go the main, the codes read the input files and create empty output file. The program reads the first line of the input file to determine the number of clients and servers. Using malloc function, we created as many empty structs as we set. The while function was used to fill empty structs. The input2 file starts after you finish the input1 file. The first line reads to determine the number of processing steps from the file. Then compare the first letter in the while loop with the letter 'o'. If the first letter is different from the letter 'o', it compares with 3 methods and starts the necessary operations. Fill the queue if the first letter goes from '3' to 'A'. Gives error code 1 if the queue is already full. The first letter fills the stack if it goes from 3 methods to 'I'. The stack is already full and gives error code 2. if the first letter goes from 3 methods to 'A', the customer deletes the stack element first. If the stack is empty, it deletes the queue element. If the structures are empty, it gives error 3. If the first letter is the same as the letter 'o', it deletes the letter on the stack. If the stack is empty, it deletes the letter in the queue. İf two structures are empty, it gives "3" (error code). Error codes fill in array in struct structure. After all the file read operations are finished, file write operations begin.The output file is populated with client and server log elements. The files are closed with the fclose method in the last section. The program ends

Stack operations in pseudo-code:

```
STACK-EMPTY(S)
  if top[S] = 0
        return true
else return false

PUSH(S, x)
  top[S] <- top[S] + 1
  S[top[S]] <- x

POP(S)
  if STACK-EMPTY(S)
        then error ”underflow”
  else top[S] <- top[S] – 1
        return S[top[S] + 1]
```

Circular queue operations in pseudo-code:

Current position = i”
Next position = (i+1)% N
previous position = (i+N-1)%N

```
Enqueue(x){
        if (rear+1)%N == front
                return
        elseif IsEmpty(){
                front ß rear ß 0}
        else{
        rear ß (rear+1)%N}
        A[rear]ß x}

Dequeue(x){
        if IsEmpty(){
        return
elseif (front == rear){
        front ß rear ß -1}
else{
        front ß (front+1)%N}
```