**linked\_stack.cpp**

**code explanation & theoretical explanation**

Circular doubly linked list is used in this code.

-Head node: used to distinguish the location of nodes and simplify insertion and deletion. The head node has no actual data.

-When the list is blank, only the head node exits. The llink and rlink of the head node points each other.

-The top element of the stack is in the right side of the head node.

-When pushing an element to the stack, insert it right next to the head node.

-When deleting an element from the stack, delete the first node which is in the right side of the head node.

1) typedef struct DlistNode

: circular doubly linked list

2) void init(DlistNode \*phead)

: initialize linked list

-phead: a pointer to the head node

3) int is\_empty(DlistNode\* phead)

: check if the list is empty

when only the head node exits, then the list is empty

- phead: a pointer to the head node

4) void push(DlistNode\* phead, element data)

: insert new node into right of head node

-phead: a pointer to the head node

-temp: node we will insert

5) element pop(DlistNode\* phead)

: delete the node which is in right side of head node and return the data

-phead: a pointer to the head node

-temp: it is a node we will delete. It is in the right side of the head node.

6) element peek(DlistNode\* phead)

: return the data of node which is in right side of head node

-phead: a pointer to the head node

**Result**

텍스트이(가) 표시된 사진

자동 생성된 설명

The result is same with the original code which was implemented using simple linked list.

**Simulation.cpp**

**code explanation & theoretical explanation**

1) int duration = 100, int max\_serv\_time = 20;

: let's set 'duration' to 100 and 'max\_serv\_time' to 20 to obtain meaningful results.

2) double random()

: Real random number generation function between 0 and 1

Use function ‘srand(time(NULL))’ in the main function to change the random numbers every time you execute the program.

3) in main()

int service\_time1 = 0;

int service\_time2 = 0;

: service\_time1 is the service time left of the customer who is receiving service from the first bank staff

service\_time2 is the service time left of the customer who is receiving service from the second bank staff

4) if (service\_time1 > 0 && service\_time2 > 0)

: when both bank staff is serving a customer

->decrement service\_time1 and service\_time2 by 1

5) if (service\_time1 > 0 && service\_time2 == 0)

: only first bank staff is serving a customer

->decrement service\_time1 by 1 & take out a customer from the queue and send him/her to the second bank staff

6) if (service\_time1 == 0 && service\_time2 > 0)

: only second bank staff is serving a customer

->decrement service\_time2 by 1 & take out a customer from the queue and send him/her to the first bank staff

7) if(service\_time1 == 0 && service\_time2 == 0)

: when no customer is receiving service

->take out a customer from the queue and start the service.

8) printf("\n");

: create empty line between the result of while loop and print\_stat()

**Result**

**텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명**

There are two bank staffs, so there are moments when two customers start service at the same time. (like when current time is 88 and 100)