1. 스택의 입출력 방식은 후입선출이다. 입출력은 맨 위에서만 일어나고 스택의 맨 밑이나 중간부터 입출력은 불가능하다.

1) 문제

1. (Programming)
The attached code ('linked_stack.cpp') was implemented using simple linked list. Re-implement the functions in the code using circular doubly linked list which is defined as below, and then run the main function of the code.

```
typedef struct DlistNode {
          element data;
          struct DlistNode *llink;
          struct DlistNode *rlink;
} DlistNode;
```

2) 문제해결

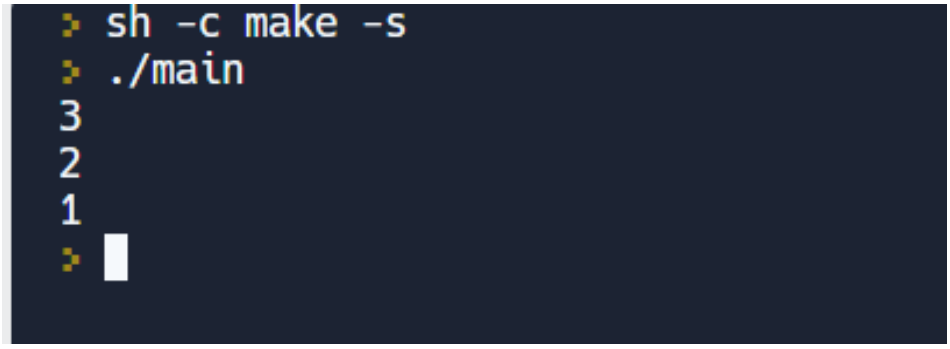
주어진 코드에선 linked stack 이 simple double linked list 로 구현되었다. 이 코드를 circular double linked list 로 바꿔본다면 structure DlistNode 에서 struct DlistNode *link 를 struct DlistNode *llink, struct DlistNode *rlink 로 바꿔줘야 한다. 스택은 top 에서만 입출력이 이뤄지기 때문에 큐와 달리 struct tail 을 만들 필요는 없다. 구조체를 정의 후 push, pop 의 함수를 입출력 할 list 를 맨 앞 list 와 rlink 와 llink 를 연결하는 식으로 수정해주면 문제를 해결 할 수 있다.

3) 구현

구조체 LinkedStackType 를 s 로 선언 후 s->top 의 값을 없애주는 식으로 list s 를 초기화해준다. Push 함수로 1 – 2 – 3 을 순서대로 집어넣어야 하는데 push 함수는 노드를 생성해 새로운 값을 노드에 넣고 스택 s 가 비어있다면 s->top 에 새 노드를 넣어준다. 비어있지 않다면 새 노드의 오른쪽 link 에 s->top 을 연결시켜주고 s->top 의 왼쪽 link 가 새 노드를 가리키게 해준다. S->top 에 새 노드를 가리키게 하고 함수를 종료한다. Push 함수로 1-2-3 을 입력했기 때문에 마지막으로 넣어준 3 이 s->top 에 있다. Pop 함수로 s->top 의 들어있는 데이터를 꺼내서 printf 로 출력해준다. 만약 s 가 비어있다면 꺼낼 값이 없으므로 종료한다. S 가 비어있지 않다면 s->top 을 새 노드로 정의해준다.(이름만 바꾼 것) 새 노드의 데이터 즉 s->top 의 데이터를 item 이라는

변수를 선언해서 넣어준다. S->top 을 s->top 의 오른쪽에 연결된 list 로 바꿔주고 새 노드를 삭제한 후 앞에서 선언하였던 item 을 반환해준다. pop 함수에 스택 s 를 넣어서 3 번 호출하기 때문에 s->top 에 있던 3 이 호출되고 s->top 이 2 로 바뀌고 2 가 호출되고 s->top 이 1 로 바뀌고 s->top 은 null 로 바뀐다.

4) 코드 실행 결과



2.

큐의 입출력 방식은 선입선출이다. 먼저 들어온 데이터가 먼저 나가고 나중에 들어온 데이터가 나중에 나가는 특성을 가진다. rear 에서 데이터 삽입이 일어나고 front 에서 데이터 삭제가 일어난다.

1) 문제

2. (Programming)
The attached code ('Simulation.cpp') was implemented with the assumption that there is a single bank staff. Revise the code accordingly in case that two bank staffs work, and then print out the results using 'print_stat()' in the code.
Note) A relatively large should be set to obtain meaningful results.

2) 문제해결

같은 clock 에서 service 를 받고 있는 지와 Service 를 받고 있고 않다면 큐에서 customer 을 꺼내와 서비스를 시작하는 행동을 2 번 반복하여 실행한다. 이때 service_time 을 2 개로 나누어서 따로 따로 체크되고 실행되게 한다. 은행원이 2 명이므로 2 개의 service_time 이 존재하는 것이다. 은행원이 2 명이 된다고 은행에 방문하는 customer 이 2 배가 되는 것은 아니므로 큐에 데이터가 삽입되는 일이 2 번 반복되진 않는다.

.

3) 구현

Service_time 2 개와 service_time 을 계산해서 각각의 service_time 에 담을 임시 service_time 을 선언하고 0 으로 초기화 시켜준다. Clock 은 0 으로 초기화 시켜주었지만 while 문 안에서 증가시켜주기 때문에 1 부터 시작한다. while 문은 설정한 duration 값과 같아질 때까지 반복된다. Clock 이 증가되고 현재시간이 표시된다. 0~1 사이에 랜덤 값이 주어지는데 그 값이 설정한 확률보다 작으면 큐에 데이터가 추가된다. 추가된 데이터는 customer 구조체의 id, 도착시간, service_time 이 저장되고 service_time 은 랜덤 값에 1 을 더한 값으로 정해진다. 이렇게 만들어진 구조체는 큐에 들어가게 된다. 은행원이 2 명이기 때문에 for 문을 돌려서 service_time 체크, 계산을 2 번 반복해줘야 한다. service_time 을 체크하고 service_time 이 0 이면 큐에서 데이터를 꺼내와야 하는데 그 전에 service_time 에 service_time1 을 넣어준다. Clock 이 증가할 때마다 for 문이 시작하기 전 반복될 것이다. Service_time 이 0 이면 큐에서 데이터를 꺼내와서 service_time 을 가져온다. Service_time 이 0 이 아니면 service_time 이 감소되고 0 이 되야 큐에서 새로운 데이터를 꺼내와 계산한다. for 문이 한번 반복됐다면 service_time1 에 service_time 을 넣고 service_time 에 service_time2 를 넣어서 2 번째 반복 때 service_time2 의 값으로 계산되게 한다. 두번째 반복 땐 service_time2 에 계산한 service_time 을 넣어준다.

코드실행결과는 다음장에 있습니다.

4) 코드 실행 결과

```
C:\Users\ssh12\CLionProjects\untitled21\cmake-build-debug\untitled21.exe
Current time=1
Customer 0 comes in 1 minutes. Service time is 3 minutes.
Customer 0 starts service in 1 minutes. Wait time was 0 minutes.
Current time=2
Customer 1 comes in 2 minutes. Service time is 5 minutes.
Customer 1 starts service in 2 minutes. Wait time was 0 minutes.
Current time=3
Customer 2 comes in 3 minutes. Service time is 3 minutes.
Current time=4
Customer 3 comes in 4 minutes. Service time is 5 minutes.
Customer 2 starts service in 4 minutes. Wait time was 1 minutes.
Current time=5
Current time=6
Current time=7
Customer 4 comes in 7 minutes. Service time is 5 minutes.
Customer 3 starts service in 7 minutes. Wait time was 3 minutes.
Customer 4 starts service in 7 minutes. Wait time was 0 minutes.
Current time=8
Current time=9
Customer 5 comes in 9 minutes. Service time is 2 minutes.
Current time=10
Customer 6 comes in 10 minutes. Service time is 1 minutes.
Current time=11
Customer 7 comes in 11 minutes. Service time is 1 minutes.
Current time=12
Customer 8 comes in 12 minutes. Service time is 5 minutes.
Customer 5 starts service in 12 minutes. Wait time was 3 minutes.
Customer 6 starts service in 12 minutes. Wait time was 2 minutes.
```

```
Current time=13
Customer 9 comes in 13 minutes. Service time is 1 minutes.
Customer 7 starts service in 13 minutes. Wait time was 2 minutes.
Current time=14
Customer 10 comes in 14 minutes. Service time is 1 minutes.
Customer 8 starts service in 14 minutes. Wait time was 2 minutes.
Customer 9 starts service in 14 minutes. Wait time was 1 minutes.
Current time=15
Customer 11 comes in 15 minutes. Service time is 3 minutes.
Customer 10 starts service in 15 minutes. Wait time was 1 minutes.
Current time=16
Customer 12 comes in 16 minutes. Service time is 4 minutes.
Customer 11 starts service in 16 minutes. Wait time was 1 minutes.
Current time=17
Customer 13 comes in 17 minutes. Service time is 1 minutes.
Current time=18
Customer 14 comes in 18 minutes. Service time is 3 minutes.
Current time=19
Customer 15 comes in 19 minutes. Service time is 1 minutes.
Customer 12 starts service in 19 minutes. Wait time was 3 minutes.
Customer 13 starts service in 19 minutes. Wait time was 2 minutes.
Current time=20
Customer 16 comes in 20 minutes. Service time is 4 minutes.
Customer 14 starts service in 20 minutes. Wait time was 2 minutes.
Number of customers served = 15
Total wait time = 23 minutes
Average wait time per person = 1.533333 minutes
Number of customers still waiting = 2
```