REPORT

과 제 명: HW#6 큐의 응용 시뮬레이션

과목명	자료구조 01분반
교수명	송오영
학번	20206319
학과	소프트웨어학부 소프트웨어전공
이름	김가연

[ServiceMan이 3명일 경우의 코드]

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define MAX_Q_SIZE 100
#define NoserviceMan 3
#define RAND_MAX 32767

int duration = 10;
double arrival_prob = 0.7;
int max_serve_time = 5;
int clocks;

int customers = 0;
int served_customers = 0;
int waited_time = 0;

typedef struct {
	int id;
	int arrival_time;
	int service_time;
}element;

typedef struct {
	element queue[MAX_Q_SIZE];
	int front;
	int rear;
}QueueType;
QueueType queue;

void init(QueueType* q) {
	q->front = q->rear = 0;
}

double random() {
	return rand() / (double)RAND_MAX;
}

int is_customer_arrived() {
	if (random() < arrival_prob) return 1;
	else return 0;
}
```

```
int is_full(QueueType* q) {
        return ((q->rear + 1) % MAX_Q_SIZE == (q->front) % MAX_Q_SIZE);
}

int is_empty(QueueType* q) {
        return (q->front == q->rear);
}

void enqueue(QueueType* q, element custom) {
        if (is_full(q)) {
                printf("큐가 꽉 찼습니다. 종료합니다.");
                exit(1);
        }
        q->rear = (q->rear + 1) % MAX_Q_SIZE;
        q->queue[q->rear] = custom;
}

element dequeue(QueueType* q) {
        if (is_empty(q)) {
                printf("큐가 비었습니다. 종료합니다.");
                exit(1);
        }
        q->front = (q->front + 1) % MAX_Q_SIZE;
        return q->queue[q->front];
}

void insert_customer(int arrival_time) {
        element customer;
        customer.id = ++customers;
        customer.arrival_time = arrival_time;
        customer.service_time = (int)(max_serve_time * random()) + 1;
        enqueue(&queue, customer);
        printf("%d id 고객이 %d분에 도착합니다. 서비스 시간은 %d입니다.\n",
customer.id, customer.arrival_time, customer.service_time);
}

int remove_customer() {
        element customer;
        int service_time = 0;

        if (is_empty(&queue)) return 0;
```

```
        customer = dequeue(&queue);
        service_time = customer.service_time - 1;
        served_customers++;
        waited_time += clocks - customer.arrival_time;
        printf("%d id 고객이 %d분에 서비스를 받기 시작하였습니다. 대기시간은 %d분 이었습니다.\n", customer.id, clocks, clocks - customer.arrival_time);
        return service_time;
}

void print_stat() {
        printf("서비스를 받은 고객 수는 %d명 입니다.\n", served_customers);
        printf("전체 대기 시간은 %d분 입니다.\n", waited_time);
        printf("1인당 평균 대기 시간은 %f분 입니다.\n", (double)waited_time / served_customers);
        printf("현재 대기중인 고객 수는 %d명 입니다.\n", customers - served_customers);
}

int main() {
        srand(time(NULL));
        int service_time[NoserviceMan] = { 0,0,0 };
        init(&queue);

        clocks = 0;

        while (clocks < duration) {
                clocks++;
                printf("현재시각 = %d\n", clocks);
                if (is_customer_arrived()) insert_customer(clocks);

                for (int i = 0; i < NoserviceMan; i++) {
                        if (service_time[i] > 0) service_time[i]--;
                        else service_time[i] = remove_customer();
                }
        }

printf("------------------------------------------------------------------------------------------\n");
        print_stat();
}
```

[결과]

```
현재시각 = 1
현재시각 = 2
현재시각 = 3
1 id 고객이 3분에 도착합니다. 서비스 시간은 2입니다.
1 id 고객이 3분에 서비스를 받기 시작하였습니다. 대기시간은 0분 이었습니다.
현재시각 = 4
현재시각 = 5
2 id 고객이 5분에 도착합니다. 서비스 시간은 5입니다.
2 id 고객이 5분에 서비스를 받기 시작하였습니다. 대기시간은 0분 이었습니다.
현재시각 = 6
3 id 고객이 6분에 도착합니다. 서비스 시간은 4입니다.
3 id 고객이 6분에 서비스를 받기 시작하였습니다. 대기시간은 0분 이었습니다.
현재시각 = 7
4 id 고객이 7분에 도착합니다. 서비스 시간은 3입니다.
4 id 고객이 7분에 서비스를 받기 시작하였습니다. 대기시간은 0분 이었습니다.
현재시각 = 8
현재시각 = 9
5 id 고객이 9분에 도착합니다. 서비스 시간은 1입니다.
현재시각 = 10
6 id 고객이 10분에 도착합니다. 서비스 시간은 4입니다.
5 id 고객이 10분에 서비스를 받기 시작하였습니다. 대기시간은 1분 이었습니다.
6 id 고객이 10분에 서비스를 받기 시작하였습니다. 대기시간은 0분 이었습니다.
--------------------------------------------------------------------------------
----------------------
서비스를 받은 고객 수는 6명 입니다.
전체 대기 시간은 1분 입니다.
1인당 평균 대기 시간은 0.166667분 입니다.
현재 대기중인 고객 수는 0명 입니다.
```

[ServiceMan이 1명일 경우의 코드]

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define MAX_Q_SIZE 100
#define NoserviceMan 1
#define RAND_MAX 32767

int duration = 10;
double arrival_prob = 0.7;
int max_serve_time = 5;
int clocks;

int customers = 0;
int served_customers = 0;
int waited_time = 0;

typedef struct {
	int id;
	int arrival_time;
	int service_time;
}element;

typedef struct {
	element queue[MAX_Q_SIZE];
	int front;
	int rear;
}QueueType;
QueueType queue;

void init(QueueType* q) {
	q->front = q->rear = 0;
}

double random() {
	return rand() / (double)RAND_MAX;
}

int is_customer_arrived() {
	if (random() < arrival_prob) return 1;
	else return 0;
}
```

```
int is_full(QueueType* q) {
        return ((q->rear + 1) % MAX_Q_SIZE == (q->front) % MAX_Q_SIZE);
}

int is_empty(QueueType* q) {
        return (q->front == q->rear);
}

void enqueue(QueueType* q, element custom) {
        if (is_full(q)) {
                printf("큐가 꽉 찼습니다. 종료합니다.");
                exit(1);
        }
        q->rear = (q->rear + 1) % MAX_Q_SIZE;
        q->queue[q->rear] = custom;
}

element dequeue(QueueType* q) {
        if (is_empty(q)) {
                printf("큐가 비었습니다. 종료합니다.");
                exit(1);
        }
        q->front = (q->front + 1) % MAX_Q_SIZE;
        return q->queue[q->front];
}

void insert_customer(int arrival_time) {
        element customer;
        customer.id = ++customers;
        customer.arrival_time = arrival_time;
        customer.service_time = (int)(max_serve_time * random()) + 1;
        enqueue(&queue, customer);
        printf("%d id 고객이 %d분에 도착합니다. 서비스 시간은 %d입니다.\n",
customer.id, customer.arrival_time, customer.service_time);
}

int remove_customer() {
        element customer;
        int service_time = 0;

        if (is_empty(&queue)) return 0;
```

```
        customer = dequeue(&queue);
        service_time = customer.service_time - 1;
        served_customers++;
        waited_time += clocks - customer.arrival_time;
        printf("%d id 고객이 %d분에 서비스를 받기 시작하였습니다. 대기시간은 %d분 이었
습니다.\n", customer.id, clocks, clocks - customer.arrival_time);
        return service_time;
}

void print_stat() {
        printf("서비스를 받은 고객 수는 %d명 입니다.\n", served_customers);
        printf("전체 대기 시간은 %d분 입니다.\n", waited_time);
        printf("1인당 평균 대기 시간은 %f분 입니다.\n", (double)waited_time /
served_customers);
        printf("현재 대기중인 고객 수는 %d명 입니다.\n", customers -
served_customers);
}

int main() {
        srand(time(NULL));
        int service_time[NoserviceMan] = { 0 };
        init(&queue);

        clocks = 0;

        while (clocks < duration) {
                clocks++;
                printf("현재시각 = %d\n", clocks);
                if (is_customer_arrived()) insert_customer(clocks);

                for (int i = 0; i < NoserviceMan; i++) {
                        if (service_time[i] > 0) service_time[i]--;
                        else service_time[i] = remove_customer();
                }
        }

printf("----------------------------------------------------------------------------
-----------------------\n");
        print_stat();
}
```

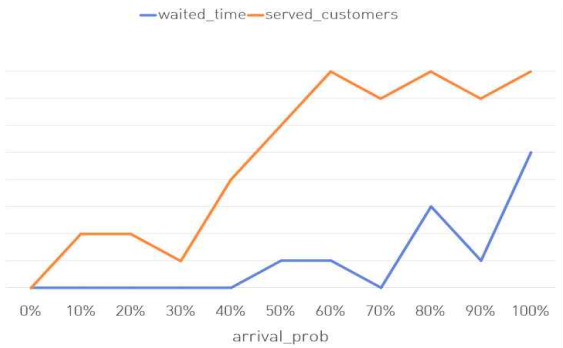
[결과]

```
현재시각 = 1
현재시각 = 2
1 id 고객이 2분에 도착합니다. 서비스 시간은 3입니다.
1 id 고객이 2분에 서비스를 받기 시작하였습니다. 대기시간은 0분 이었습니다.
현재시각 = 3
2 id 고객이 3분에 도착합니다. 서비스 시간은 4입니다.
현재시각 = 4
현재시각 = 5
3 id 고객이 5분에 도착합니다. 서비스 시간은 1입니다.
2 id 고객이 5분에 서비스를 받기 시작하였습니다. 대기시간은 2분 이었습니다.
현재시각 = 6
4 id 고객이 6분에 도착합니다. 서비스 시간은 5입니다.
현재시각 = 7
5 id 고객이 7분에 도착합니다. 서비스 시간은 5입니다.
현재시각 = 8
6 id 고객이 8분에 도착합니다. 서비스 시간은 3입니다.
현재시각 = 9
7 id 고객이 9분에 도착합니다. 서비스 시간은 3입니다.
3 id 고객이 9분에 서비스를 받기 시작하였습니다. 대기시간은 4분 이었습니다.
현재시각 = 10
8 id 고객이 10분에 도착합니다. 서비스 시간은 2입니다.
4 id 고객이 10분에 서비스를 받기 시작하였습니다. 대기시간은 4분 이었습니다.
--------------------------------------------------------------------------------
----------------------
서비스를 받은 고객 수는 4명 입니다.
전체 대기 시간은 10분 입니다.
1인당 평균 대기 시간은 2.500000분 입니다.
현재 대기중인 고객 수는 4명 입니다.
```

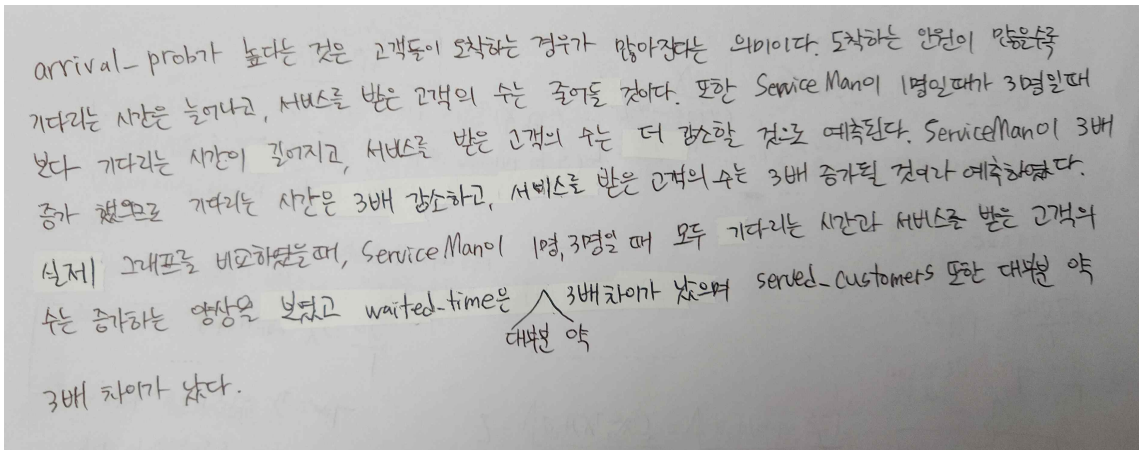
[그래프 및 검토의견]

case 1. arrival_prob에 따른 customer의 waited_time과 served_customers에 대한 비교 (max_serve_time = 5)

<ServiceMan 1명일때>

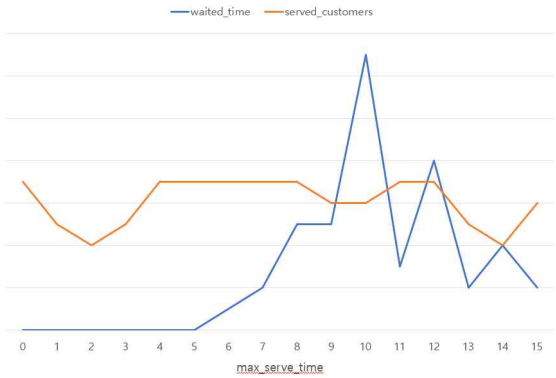


<ServiceMan 3명일때>



case 2. max_serve_time에 따른 customer의 waited_time과 served_customers에 대한 비교 (arrival_prob = 0.7)

<ServiceMan 1명일때>



<ServiceMan 3명일때>

max_serve_time이 증가하면 대기 시간은 길어지고 그에 따라 서비스를 받은 고객의 수는 줄어들 것이다.
또한 ServiceMan이 1명일때가 3명일때 보다 대기 시간이 더 길어지고, 서비스를 받은 고객의 수는 더 줄을 것이다.
ServiceMan의 수가 3배 차이나므로 1명일때와 3명일때의 각 수치가 3배 차이 날 것이라 예측하였다.
실제 데이터를 바탕으로 그려진 그래프를 보면, waited_time은 3명일때 대체로 줄며 served_customers은 많아졌다. 대체로 waited_time 과 served_customers 은 ServiceMan이 1명일때와 3명일때, 3배 차이 났다.