

자료구조

- 과제번호2 -

학번: 20202878

이름: 조성주

1. 스택의 원소를 거꾸로 저장하는 프로그램(연결리스트를 이용한 3개의 스택 응용)

1-A. 소스 코드

```
#include <stdio.h>

#include <stdlib.h>

typedef int element;    //스택 원소(element)의 자료형을 int로 정의

typedef struct stackNode {    //스택의 노드를 구조체로 정의
    element data;

    struct stackNode *link;
} stackNode;

void push(element item, stackNode **top1) {
    stackNode* temp = (stackNode*) malloc(sizeof(stackNode));

    temp->data = item;

    temp->link = *top1;    //삽입 노드를 top의 위에 연결

    *top1 = temp;    //top 위치를 삽입 노드로 이동
}

element pop(stackNode **top1) {
    element item;

    stackNode* temp = *top1;

    item = temp->data;

    *top1 = (*top1)->link;

    free(temp);    //temp 공간 해제

    return item;
}
```

```

void reverse_stack(stackNode **top0) {

    stackNode *top1 = NULL;

    stackNode *top2 = NULL;


    while(*top0 != NULL) {

        push(pop(top0), &top1);

    }

    while(top1 != NULL) {

        push(pop(&top1), &top2);

    }

    while(top2 != NULL) {

        push(pop(&top2), top0);

    }

}

```

```

void printStack(stackNode **top) {

    stackNode* p = *top;

    printf("STACK [ ");

    while (p) {

        printf("%d ", p->data);

        p = p->link;

    }

    printf("] ");

}

```

```

int main() {

    int data;

    stackNode *top = NULL;

    push(1, &top);

    push(2, &top);

    push(3, &top);


    reverse_stack(&top);    //push, pop 과정을 함축

    printStack(&top);      //스택의 원소를 top에서 bottom 순서로 출력

    return 0;

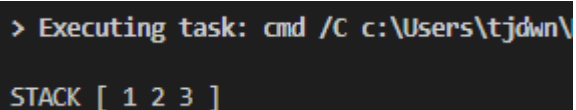
}

```

1-B. 소스 코드 실행 방법

컴파일 후 실행

1-C. 출력 결과



```

> Executing task: cmd /C c:\Users\tjdown\
STACK [ 1 2 3 ]

```

2. 중위표기식을 후위표기식으로 변환, 계산하는 프로그램(스택 이용)

2-A. 소스 코드

```

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX_STACK_SIZE 100

```

```
typedef char element;

element stack[MAX_STACK_SIZE];

int top;
```

```
void init_stack() {

    top = -1;

}
```

```
int is_Empty() {

    return (top == -1);

}
```

```
void push(element item) {

    stack[++top] = item;

}
```

```
element pop() {

    return stack[top--];

}
```

```
element peek() {

    return stack[top];

}
```

```
//연산자의 우선순위 반환
```

```

int prec(char op) {
    switch(op) {
        case '(': case ')':
            return 0;

        case '+': case '-':
            return 1;

        case '*': case '/':
            return 2;
    }

    return -1;
}

```

//수식 변환

```

element* infix_to_postfix(char exp[]) {
    int i, idx = 0;

    char ch, op;

    int len = strlen(exp);

    element* postfix_arr = (element*)malloc(MAX_STACK_SIZE);

    init_stack();

    for (i = 0; i < len; i++) {
        ch = exp[i];

        //일반 숫자의 경우
        if ('0' <= ch && ch <= '9') {
            postfix_arr[idx++] = ch;

```

```

    }

    //연산자 +, -, *, /의 경우
    else if (ch == '+' || ch == '-' || ch == '*' || ch == '/') {
        while (!is_Empty() && (prec(ch) <= prec(peek()))) {
            postfix_arr[idx++] = peek();

            pop();
        }

        push(ch);
    }

    //'('의 경우 무조건 스택에 추가
    else if (ch == '(') {
        push(ch);
    }

    //')'의 경우 '('가 나올 때까지 스택에서 pop하여 추가
    else if (ch == ')') {
        op = pop();

        while (op != '(') {
            postfix_arr[idx++] = op;

            op = pop();
        }
    }
}

```

```

while (!is_Empty()) {    //스택에 저장된 연산자들 출력

    op = peek();

    pop();
}

```

```

        postfix_arr[idx++] = op;
    }

    postfix_arr[idx] = '\0';        //문자열의 끝 지정

    return postfix_arr;
}

```

//후위 표기법 수식 계산

```

int evalPostfix(char exp[]) {

    int opr1, opr2, value, i = 0;

    int length = strlen(exp);

    char symbol;

    init_stack();

    for (i = 0; i < length; i++) {

        symbol = exp[i];

        if (symbol != '+' && symbol != '-' && symbol != '*' && symbol != '/') {

            value = symbol - '0';    //char형 숫자로 변환

            push(value);

        }

        else {

            opr2 = pop();

            opr1 = pop();

            switch(symbol) {        //변수 opr1과 opr2에 대해 symbol에 저장된 연산자를 연산

                case '+': push(opr1 + opr2); break;

                case '-': push(opr1 - opr2); break;

                case '*': push(opr1 * opr2); break;

```



```

        case '/': push(opr1 / opr2); break;
    }
}

}

return pop();    //수식에 대한 처리를 마친 후 스택에 남아있는 결과값을 반환
}

int main() {
    char expr[MAX_STACK_SIZE];

    printf("중위 표기식: ");
    scanf("%s", expr);
    element *result = infix_to_postfix(expr);
    printf("후위표기식: %s\n", result);
    printf("계산 값: %d\n", evalPostfix(result));

    return 0;
}

```

2-B. 소스 코드 실행 방법

컴파일 후 실행

2-C. 출력 결과

```
> Executing task: cmd /C c:\Users\tjdwn\l  
중위 표기식: 2-6/3*6+4  
후위 표기식: 263/6*-4+  
계산 값: -6
```

3. 중위표기식을 전위표기식으로 변환 프로그램

3-A. 소스 코드

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

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

```
#include <string.h>
```

```
#define MAX_STACK_SIZE 100
```

```
typedef char element;
```

```
element stack[MAX_STACK_SIZE];
```

```
int top;
```

```
void init_stack() {
```

```
    top = -1;
```

```
}
```

```
int is_Empty() {
```

```
    return(top == -1);
```

```
}
```

```
int is_Full() {
```

```
    return(top) == (MAX_STACK_SIZE - 1);  
}
```

```
void push(element item) {
```

```
    stack[++top] = item;  
}
```

```
element pop() {  
    return stack[top--];  
}
```

```
element peek() {  
    return stack[top];  
}
```

//연산자의 우선순위 반환

```
int prec(char op) {  
    switch(op) {  
        case '(': case ')':  
            return 0;  
        case '+': case '-':  
            return 1;  
        case '*': case '/':  
            return 2;  
    }  
}
```

```
    return -1;
}
```

//수식을 역순으로 변경

```
void reverse_exp(char exp[]) {
```

```
    int i = 0, j = 0;
```

```
    int len = strlen(exp);
```

```
    char *a = (char*)malloc(sizeof(char) * (len + 1));
```

```
    init_stack();
```

```
    //exp에 들어있는 요소를 배열 a에 옮김
```

```
    for (i = 0; i < len; i++) {
```

```
        a[i] = exp[i];
```

```
    }
```

```
    //스택에 배열 a의 요소들 push
```

```
    for (i = 0; i < len; i++) {
```

```
        if (a[i] == '(')
```

```
            push('(');
```

```
        else if (a[i] == ')')
```

```
            push('(');
```

```
        else
```

```
            push(a[i]);
```

```
    }
```

```
    //exp에 스택에 있는 요소들을 pop하여 나온 원소들 삽입
```

```

    for (i = 0; i < len; i++) {

        exp[i] = pop();

    }

}

```

//수식 변환

```

element* infix_to_postfix(char exp[]) {

    int i, idx = 0;

    char ch, op;

    int len = strlen(exp);

    element* postfix_arr = (element*)malloc(MAX_STACK_SIZE);

    init_stack();

    for (i = 0; i < len; i++) {

        ch = exp[i];

        //일반 숫자의 경우

        if ('0' <= ch && ch <= '9') {

            postfix_arr[idx++] = ch;

        }

        //연산자 +, -, *, /의 경우

        else if (ch == '+' || ch == '-' || ch == '*' || ch == '/') {

            while (!is_Empty() && (prec(ch) <= prec(peek()))) {

                postfix_arr[idx++] = peek();

                pop();

            }

```

```

        push(ch);
    }

    //'('의 경우 무조건 스택에 추가

    else if (ch == '(') {
        push(ch);
    }

    //'')의 경우 '('가 나올 때까지 스택에서 pop하여 추가

    else if (ch == ')') {
        op = pop();

        while (op != '(') {
            postfix_arr[idx++] = op;

            op = pop();
        }
    }
}

while (!is_Empty()) {    //스택에 저장된 연산자들 출력

    op = peek();

    pop();

    postfix_arr[idx++] = op;
}

postfix_arr[idx] = '\0';    //문자열의 끝 지정

return postfix_arr;
}

int main() {

```

```

char expr[MAX_STACK_SIZE];

printf("중위 표기식: ");

scanf("%s", expr);

reverse_exp(expr);      //역순 변경

element *result = infix_to_postfix(expr);      //후위표기식으로 변환

reverse_exp(result);      //후위표기식을 역순으로 변경

printf("전위 표기식: %s\n", result);

return 0;

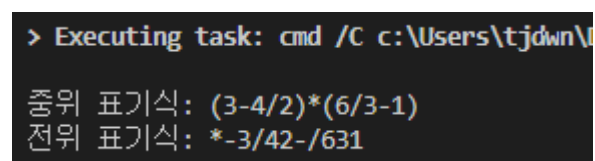
}

```

3-B. 소스 코드 실행 방법

컴파일 후 실행

3-C. 출력 결과



```

> Executing task: cmd /C c:\Users\tjdown\
중위 표기식: (3-4/2)*(6/3-1)
전위 표기식: *-3/42-/631

```

4. 스택 이용 미로 구현, 입구에서 출구로 가는 경로 출력(8방향 탐색)

4-A. 소스 코드

```

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

```

```
#define MAX_STACK_SIZE 100
```

```
#define EXIT_ROW 14
```

```
#define EXIT_COL 15
```

```
typedef struct {
```

```
    short int x;
```

```
    short int y;
```

```
} offsets;
```

```
offsets move[8] = {{-1,0}, {-1,1}, {0,1}, {1,1}, {1,0}, {1,-1}, {0,-1}, {-1,-1}};
```

```
typedef struct {
```

```
    short int row;
```

```
    short int col;
```

```
    short int dir;
```

```
} element;
```

```
element stack[MAX_STACK_SIZE];
```

```
//미로 생성
```

```
int maze[EXIT_ROW+2][EXIT_COL+2] = {
```

```
    {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},
```

```
    {1,0,0,0,1,0,0,1,1,0,0,0,0,1,0,0,1},
```

```
    {1,0,1,0,1,0,0,0,0,0,1,0,0,0,0,1,1},
```

```
    {1,0,1,1,1,0,0,1,0,1,1,1,0,1,0,1,1},
```

```
    {1,0,0,0,1,0,0,0,0,0,1,0,0,1,0,0,1},
```

```
    {1,0,1,0,0,0,1,1,1,0,1,0,1,1,1,0,1},
```



```

        {1,0,1,0,0,1,1,0,1,0,1,0,0,0,0,1},
        {1,0,1,1,1,1,0,0,1,0,0,1,0,1,1,0,1},
        {1,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0},
        {1,0,1,1,0,1,0,0,0,1,0,0,1,1,0,0,1},
        {1,1,0,1,0,0,0,1,0,0,0,0,0,0,0,0,1},
        {1,0,0,1,0,0,1,0,1,0,0,1,0,0,1,1,1},
        {1,1,0,0,0,1,0,0,1,1,1,0,0,0,0,0,1},
        {1,1,0,1,1,0,0,0,0,1,0,0,0,1,1,0,1},
        {1,0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,1},
        {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},
};

int mark[EXIT_ROW+2][EXIT_COL+2];

int top;

void push(element item) {

    stack[++top] = item;

}

element pop() {

    return stack[top--];

}

element peek() {

    return stack[top];

}

```

```

void path() {

    int i, row, col, next_row, next_col, dir;

    int found = 0;

    element position;

    mark[1][1] = 1;

    top = 0;

    stack[0].row = 1;

    stack[0].col = 1;

    stack[0].dir = 2;

    while (top > -1 && !found) {

        position = pop();

        row = position.row;

        col = position.col;

        dir = position.dir;

        while (dir < 8 && !found) {

            next_row = row + move[dir].x;

            next_col = col + move[dir].y;

            if (next_row == EXIT_ROW && next_col == EXIT_COL)

                found = 1;

            else if (!maze[next_row][next_col] && !mark[next_row][next_col]) {

                mark[next_row][next_col] = 1;

                position.row = row;

                position.col = col;

                position.dir = ++dir;

```

```

        push(position);

        //갱신

        row = next_row;

        col = next_col;

        dir = 0;

    }

    else

        ++dir;

}

}

```

```

if (found) {

    printf("The path is:\n");

    printf("row col\n");

    for (i = 0; i <= top; i++)

        printf("%2d%5d", stack[i].row, stack[i].col);    //스택 경로 출력

    printf("%2d%5d\n", row, col);

    printf("%2d%5d\n", EXIT_ROW, EXIT_COL);

}

else printf("The maze does not have a path\n");

}

```

```

int main() {

    path();

    return 0;

}

```

4-B. 소스 코드 실행 방법

컴파일 후 실행

4-C. 출력 결과

```
> Executing task: cmd /C c:\Users\tjdown\Desktop\SSU20\3_1\DataStructure\과제번호2_2  
0202878\Assignment2_4 <
```

The path is:

row col

1	1 1	2 2	1 3	1 4	2 4	3 5	4 4	5 3	5 2	5 1	5 1
6 2	7 2	8 1	9 1	10 1	11 1	12 2	13 1	14 1	15 2	14 3	14 4
15 5	15 6	15 7	15 8	16 9	15 8	15 9	14 8	14 8	13 8	12 7	12 6
12 6	11 7	10 8	10 9	11 10	12 10	13 10	14 11	13 12	14 12	15 13	15
14	15										

5. 큐를 이용 은행 시뮬레이션 작성, 고객들의 총 대기시간 출력

5-A. 소스 코드

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

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

```
#include <stdbool.h>
```

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

```
#define MAX_QUEUE_SIZE 100
```

```
typedef struct {
```

```
    int id;
```

```
    int arrival_time;
```

```
    int service_time;
```

```
} element;
```

```
typedef struct {  
    int front;  
    int rear;  
    element data[MAX_QUEUE_SIZE];  
} QueueType;
```

```
void init_queue(QueueType *q) {  
    q->rear = -1;  
    q->front = -1;  
}
```

```
int is_empty(QueueType *q) {  
    if (q->front == q->rear)  
        return 1;  
    else return 0;  
}
```

```
int is_full(QueueType *q) {  
    if (q->rear == MAX_QUEUE_SIZE - 1)  
        return 1;  
    else return 0;  
}
```

```
void enqueue(QueueType *q, element item) {  
    if (is_full(q)) {
```

```
        printf("큐가 포화상태");

        exit(1);

    }

    q->data[++(q->rear)] = item;
}
```

```
element dequeue(QueueType *q) {

    if (is_empty(q)) {

        printf("큐가 공백상태");

        exit(1);

    }

    element item = q->data[++(q->front)];

    return item;

}
```

```
element peek(QueueType *q) {

    if (is_empty(q))

        exit(1);

    else

        return q->data[q->front + 1];

}
```

```
int bank_simul() {

    int minutes = 60;

    int total_wait = 0;

    int total_customers = 0;
```

```

int a_service_time = 0;

int b_service_time = 0;

int a_service_customer;

int b_service_customer;

bool aCounter = true;

bool bCounter = true;

QueueType q;

init_queue(&q);

srand(time(NULL));


for (int clock = 0; clock < minutes; clock++) {

    printf("현재시각 = %d\n", clock);

    if ((rand() % 10) < 3) {

        element customer;

        customer.id = total_customers++;

        customer.arrival_time = clock;

        customer.service_time = rand() % 3 + 1;

        enqueue(&q, customer);

        printf("고객 %d이 %d분에 들어옵니다. 업무처리시간 = %d분\n", customer.id,
customer.arrival_time, customer.service_time);

    }


    if (a_service_time > 0) {

        printf("고객 %d이 A창구에서 업무처리중입니다.\n", a_service_customer);

        a_service_time--;

        if (a_service_time == 0) {

```

```

        aCounter = true;
    }
}

else if (aCounter) {
    if (!is_empty(&q)) {
        element customer = dequeue(&q);

        a_service_customer = customer.id;

        a_service_time = customer.service_time;

        printf("고객 %d이 %d분에 A창구에서 업무를 시작합니다. 대기시간은 %d분이었
습니다.\n", customer.id, clock, clock - customer.arrival_time);

        aCounter = false;

        total_wait += clock - customer.arrival_time;
    }
}

if (b_service_time > 0) {
    printf("고객 %d이 B창구에서 업무처리중입니다.\n", b_service_customer);

    b_service_time--;

    if (b_service_time == 0) {
        bCounter = true;
    }
}

else if (!is_empty(&q)) {
    element customer = dequeue(&q);

    b_service_customer = customer.id;

    b_service_time = customer.service_time;

```



```
printf("고객 %d이 %d분에 B창구에서 업무를 시작합니다. 대기시간은 %d분이었습니다.\n", customer.id, clock, clock - customer.arrival_time);
```

```
bCounter = false;
```

```
total_wait += clock - customer.arrival_time;
```

```
}
```

```
}
```

```
printf("전체 대기 시간 = %d분 \n", total_wait);
```

```
return 0;
```

```
}
```

```
int main() {
```

```
bank_simul();
```

```
return 0;
```

```
}
```

5-B. 소스 코드 실행 방법

컴파일 후 실행

5-C. 출력 결과

```
> Executing task: cmd /C c:\Users\tjdown\Desktop\SSU20\3_1\DataStructure\과제번호2_20202878\Assignment2_5 <
현재시각 = 0
현재시각 = 1
현재시각 = 2
현재시각 = 3
고객 0이 3분에 들어옵니다. 업무처리시간 = 1분
고객 0이 3분에 A창구에서 업무를 시작합니다. 대기시간은 0분이었습니다.
현재시각 = 4
고객 0이 A창구에서 업무처리중입니다.
현재시각 = 5
현재시각 = 6
현재시각 = 7
현재시각 = 8
```

고객 10이 8분에 들어옵니다. 업무처리시간 = 3분
 고객 10이 8분에 A창구에서 업무를 시작합니다. 대기시간은 0분이었습니다.
 현재시각 = 9
 고객 20이 9분에 들어옵니다. 업무처리시간 = 1분
 고객 10이 A창구에서 업무처리중입니다.
 고객 20이 9분에 B창구에서 업무를 시작합니다. 대기시간은 0분이었습니다.
 현재시각 = 10
 고객 10이 A창구에서 업무처리중입니다.
 고객 20이 B창구에서 업무처리중입니다.
 현재시각 = 11
 고객 30이 11분에 들어옵니다. 업무처리시간 = 2분
 고객 10이 A창구에서 업무처리중입니다.
 고객 30이 11분에 B창구에서 업무를 시작합니다. 대기시간은 0분이었습니다.
 현재시각 = 12
 고객 40이 12분에 들어옵니다. 업무처리시간 = 2분
 고객 40이 12분에 A창구에서 업무를 시작합니다. 대기시간은 0분이었습니다.
 고객 30이 B창구에서 업무처리중입니다.
 현재시각 = 13
 고객 40이 A창구에서 업무처리중입니다.
 고객 30이 B창구에서 업무처리중입니다.
 현재시각 = 14
 고객 50이 14분에 들어옵니다. 업무처리시간 = 3분
 고객 40이 A창구에서 업무처리중입니다.
 고객 50이 14분에 B창구에서 업무를 시작합니다. 대기시간은 0분이었습니다.
 현재시각 = 15
 고객 60이 15분에 들어옵니다. 업무처리시간 = 2분
 고객 60이 15분에 A창구에서 업무를 시작합니다. 대기시간은 0분이었습니다.
 고객 50이 B창구에서 업무처리중입니다.
 현재시각 = 16
 고객 60이 A창구에서 업무처리중입니다.
 고객 50이 B창구에서 업무처리중입니다.
 현재시각 = 17
 고객 60이 A창구에서 업무처리중입니다.
 고객 50이 B창구에서 업무처리중입니다.
 현재시각 = 18
 현재시각 = 19
 고객 70이 19분에 들어옵니다. 업무처리시간 = 1분
 고객 70이 19분에 A창구에서 업무를 시작합니다. 대기시간은 0분이었습니다.
 현재시각 = 20
 고객 70이 A창구에서 업무처리중입니다.
 현재시각 = 21
 현재시각 = 22
 고객 80이 22분에 들어옵니다. 업무처리시간 = 3분
 고객 80이 22분에 A창구에서 업무를 시작합니다. 대기시간은 0분이었습니다.
 현재시각 = 23
 고객 80이 A창구에서 업무처리중입니다.
 현재시각 = 24
 고객 80이 A창구에서 업무처리중입니다.
 현재시각 = 25
 고객 80이 A창구에서 업무처리중입니다.
 현재시각 = 26
 고객 90이 26분에 들어옵니다. 업무처리시간 = 1분
 고객 90이 26분에 A창구에서 업무를 시작합니다. 대기시간은 0분이었습니다.
 현재시각 = 27

현재시각 = 28
 고객 100이 28분에 들어옵니다. 업무처리시간 = 3분
 고객 100이 28분에 A창구에서 업무를 시작합니다. 대기시간은 0분이었습니다.
 현재시각 = 29
 고객 100이 A창구에서 업무처리중입니다.
 현재시각 = 30
 고객 110이 30분에 들어옵니다. 업무처리시간 = 3분
 고객 100이 A창구에서 업무처리중입니다.
 고객 110이 30분에 B창구에서 업무를 시작합니다. 대기시간은 0분이었습니다.
 현재시각 = 31
 고객 120이 31분에 들어옵니다. 업무처리시간 = 3분
 고객 100이 A창구에서 업무처리중입니다.
 고객 110이 B창구에서 업무처리중입니다.
 현재시각 = 32
 고객 120이 32분에 A창구에서 업무를 시작합니다. 대기시간은 1분이었습니다.
 고객 110이 B창구에서 업무처리중입니다.
 현재시각 = 33

고객 130이 33분에 들어옵니다. 업무처리시간 = 3분
고객 120이 A창구에서 업무처리중입니다.
고객 110이 B창구에서 업무처리중입니다.
현재시각 = 34
고객 120이 A창구에서 업무처리중입니다.
고객 130이 34분에 B창구에서 업무를 시작합니다. 대기시간은 1분이었습니다.
현재시각 = 35
고객 120이 A창구에서 업무처리중입니다.
고객 130이 B창구에서 업무처리중입니다.
현재시각 = 36
고객 140이 36분에 들어옵니다. 업무처리시간 = 2분
고객 140이 36분에 A창구에서 업무를 시작합니다. 대기시간은 0분이었습니다.
고객 130이 B창구에서 업무처리중입니다.
현재시각 = 37
고객 150이 37분에 들어옵니다. 업무처리시간 = 2분
고객 140이 A창구에서 업무처리중입니다.
고객 130이 B창구에서 업무처리중입니다.
현재시각 = 38
고객 140이 A창구에서 업무처리중입니다.
고객 150이 38분에 B창구에서 업무를 시작합니다. 대기시간은 1분이었습니다.
현재시각 = 39
고객 150이 B창구에서 업무처리중입니다.
현재시각 = 40
고객 150이 B창구에서 업무처리중입니다.
현재시각 = 41
고객 160이 41분에 들어옵니다. 업무처리시간 = 1분
고객 160이 41분에 A창구에서 업무를 시작합니다. 대기시간은 0분이었습니다.
현재시각 = 42
고객 160이 A창구에서 업무처리중입니다.
현재시각 = 43
현재시각 = 44
고객 170이 44분에 들어옵니다. 업무처리시간 = 1분
고객 170이 44분에 A창구에서 업무를 시작합니다. 대기시간은 0분이었습니다.
현재시각 = 45
고객 170이 A창구에서 업무처리중입니다.
현재시각 = 46
현재시각 = 47
현재시각 = 48
고객 180이 48분에 들어옵니다. 업무처리시간 = 3분
고객 180이 48분에 A창구에서 업무를 시작합니다. 대기시간은 0분이었습니다.
현재시각 = 49
고객 180이 A창구에서 업무처리중입니다.
현재시각 = 50
고객 180이 A창구에서 업무처리중입니다.
현재시각 = 51
고객 180이 A창구에서 업무처리중입니다.
현재시각 = 52
현재시각 = 53
현재시각 = 54
현재시각 = 55
현재시각 = 56
현재시각 = 57
현재시각 = 58
현재시각 = 59
전체 대기 시간 = 3분