* **HW3\_1\_1**

int get\_count(QueueType \*q)

{

int count;

count = q->rear - q->front;

return count;

}

int get\_count(QueueType \*q)

{

int count = 0;

int i;

for (i = q->front+1; i <= q->rear; i++) {

count++;

}

}

int get\_count(QueueType \*q)

{

if (is\_empty(q))

return 0;

else if (q->front < q->rear)

return q->rear - q->front;

else

return MAX\_QUEUE\_SIZE - q->front + q->rear;

}

return count;

}

// best

int get\_count(QueueType \*q)

{

if (q->front <= q->rear)

return (q->rear - q->front)

else

return (q->rear + MAX\_QUEUE\_SIZE - q->front)

}

* **HW3\_1\_1**

void print\_queue(QueueType \*q)

{

int i;

for(i=q->front+1;i<=q->rear;i++)

printf("%s ", q->queue[i] );

printf("\n");

}

void print\_queue(QueueType \*q)

{

int i;

int start = (q->front + 1) % MAX\_QUEUE\_SIZE;

int end = (q->rear + 1) % MAX\_QUEUE\_SIZE;

// 인덱스에 +1 을 하면 MAX\_QUEUE\_SIZE를 초과할 수 있기 때문에 다시 % 연산을 해준다.

**if (!(is\_empty(q))) {**

for (i = start; i != end; i = (i + 1) % MAX\_QUEUE\_SIZE)

printf("%s ", q->queue[i].name);

**}**

printf("\n");

}

// best

void print\_queue(QueueType \*q)

{

int temp = (q->front + 1) % MAX\_QUEUE\_SIZE;

while (temp != (q->rear + 1) % MAX\_QUEUE\_SIZE) {

printf("%s ", q->queue[temp].name);

temp = (temp + 1) % MAX\_QUEUE\_SIZE;

}

printf("\n");

}