

TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com

학생 – GJ (박현우)
uc820@naver.com

1.큐(Queue) 코드 이해(그림 그리기) - (enqueue 1)

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

```
#define EMPTY 0
```

```
struct node{
```

```
int data;
struct node *link;
};
```

```
typedef struct node Queue;
```

```
Queue *getnode(){
Queue *tmp;
tmp = (Queue *)malloc(sizeof(Queue));
tmp->link = EMPTY;
return tmp;
}
```

```
void dequeue(Queue **head, int data ){
```

```
Queue *tmp;
if( *head == EMPTY){

printf("QUEUE IS EMPTY\n");
}
```

```
tmp = *head;
if(tmp->data ==data){

*head = tmp->link;
free(tmp);
}else
dequeue( &(tmp->link), data);
}
```

```
void enqueue(Queue **head, int data){
if( *head == EMPTY){
```

```
*head = getnode();
(*head)->data = data;
return;
}
enqueue( &((*head)->link), data);
}
```

```
void print_queue(Queue *head){

Queue *tmp = head;

while(tmp){

printf("%d\n",tmp->data);
tmp = tmp->link;
}
}
```

```
int main(void){

Queue *head = EMPTY;

enqueue(&head, 10);
enqueue(&head, 20);
enqueue(&head, 30);

print_queue(head);

dequeue(&head,20);

print_queue(&head);

return 0;
}
```

Stack

```
main
0 = EMPTY
head 1000 : tmp주소
enqueue
1000 head 10 data
get_node
2000 tmp
```

Heap

```
data link
0 = EMPTY
2000 : tmp 주소
```

1.큐(Queue) 코드 이해(그림 그리기) - (enqueue 2)

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

```
#define EMPTY 0
```

```
struct node{
```

```
int data;
struct node *link;
};
```

```
typedef struct node Queue;
```

```
Queue *getnode(){
Queue *tmp;
tmp = (Queue *)malloc(sizeof(Queue));
tmp->link = EMPTY;
return tmp;
}
```

```
void dequeue(Queue **head, int data ){
```

```
Queue *tmp;
if( *head == EMPTY){

printf("QUEUE IS EMPTY\n");
}
```

```
tmp = *head;
if(tmp->data ==data){
```

```
*head = tmp->link;
free(tmp);
}else
dequeue( &(tmp->link), data);
}
```

```
void enqueue(Queue **head, int data){
if( *head == EMPTY){
```

```
*head = getnode();
(*head)->data = data;
return;
}
enqueue( &((*head)->link), data);
}
```

```
void print_queue(Queue *head){
```

```
Queue *tmp = head;
```

```
while(tmp){

printf("%d\n",tmp->data);
tmp = tmp->link;
}
```

```
}
```

```
int main(void){
```

```
Queue *head = EMPTY;
```

```
enqueue(&head, 10);
enqueue(&head, 20);
enqueue(&head, 30);
```

```
print_queue(head);
```

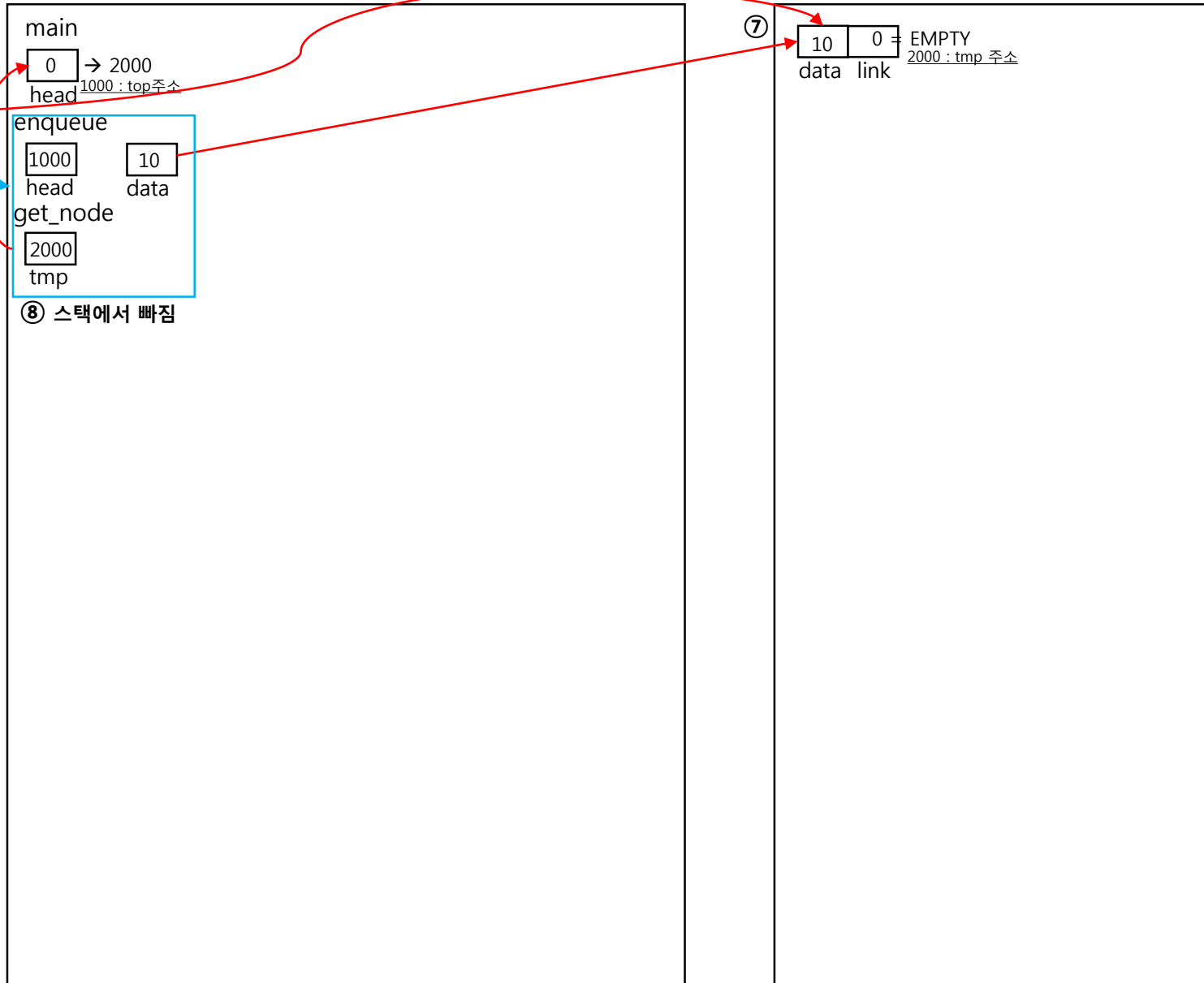
```
dequeue(&head,20);
```

```
print_queue(&head);
```

```
return 0;
}
```

Stack

Heap



1.큐(Queue) 코드 이해(그림 그리기) - (enqueue 3)

Stack

Heap

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

```
#define EMPTY 0
```

```
struct node{
```

```
int data;
struct node *link;
};
```

```
typedef struct node Queue;
```

```
Queue *getnode(){
Queue *tmp;
tmp = (Queue *)malloc(sizeof(Queue));
tmp->link = EMPTY;
return tmp;
}
```

```
void dequeue(Queue **head, int data ){
```

```
Queue *tmp;
if( *head == EMPTY){

printf("QUEUE IS EMPTY\n");
}
```

```
tmp = *head;
if(tmp->data ==data){
```

```
*head = tmp->link;
free(tmp);
}else
dequeue( &(amp;tmp->link), data);
}
```

```
void enqueue(Queue **head, int data){
if( *head == EMPTY){
```

```
*head = getnode();
(*head)->data = data;
return;
}
enqueue( &((*head)->link), data);
}
```

```
void print_queue(Queue *head){
```

```
Queue *tmp = head;
```

```
while(tmp){
printf("%d\n",tmp->data);
tmp = tmp->link;
}
```

```
}
```

```
int main(void){
```

```
Queue *head = EMPTY;
```

```
enqueue(&head, 10);
enqueue(&head, 20);
enqueue(&head, 30);
```

```
print_queue(head);
```

```
dequeue(&head,20);
```

```
print_queue(&head);
```

```
return 0;
}
```

main

0 → 2000
head → 1000 : top주소

enqueue

1000 20
head data

enqueue

2004 20
head data

get_node

3000
tmp

10 0 EMPTY
data link 2000 : tmp 주소
 2004 : link 주소

 3000 : tmp 주소
 3004 : link 주소

1. 큐(Queue) 코드 이해(그림 그리기) - (enqueue 4)

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

#define EMPTY 0

struct node{
int data;
struct node *link;
};

typedef struct node Queue;

Queue *getnode(){
Queue *tmp;
tmp = (Queue *)malloc(sizeof(Queue));
tmp->link = EMPTY;
return tmp;
}

void dequeue(Queue **head, int data ){
Queue *tmp;
if( *head == EMPTY){
printf("QUEUE IS EMPTY\n");
}

tmp = *head;
if(tmp->data ==data){
*head = tmp->link;
free(tmp);
}else
dequeue( &(tmp->link), data);
}
```

```
void enqueue(Queue **head, int data){
    if( *head == EMPTY){
        *head = getnode();
        (*head)->data = data;
        return;
    }
    enqueue( &((*head)->link), data);
}

void print_queue(Queue *head){
    Queue *tmp = head;

    while(tmp){
        printf("%d\\n",tmp->data);
        tmp = tmp->link;
    }
}

int main(void){
    Queue *head = EMPTY;

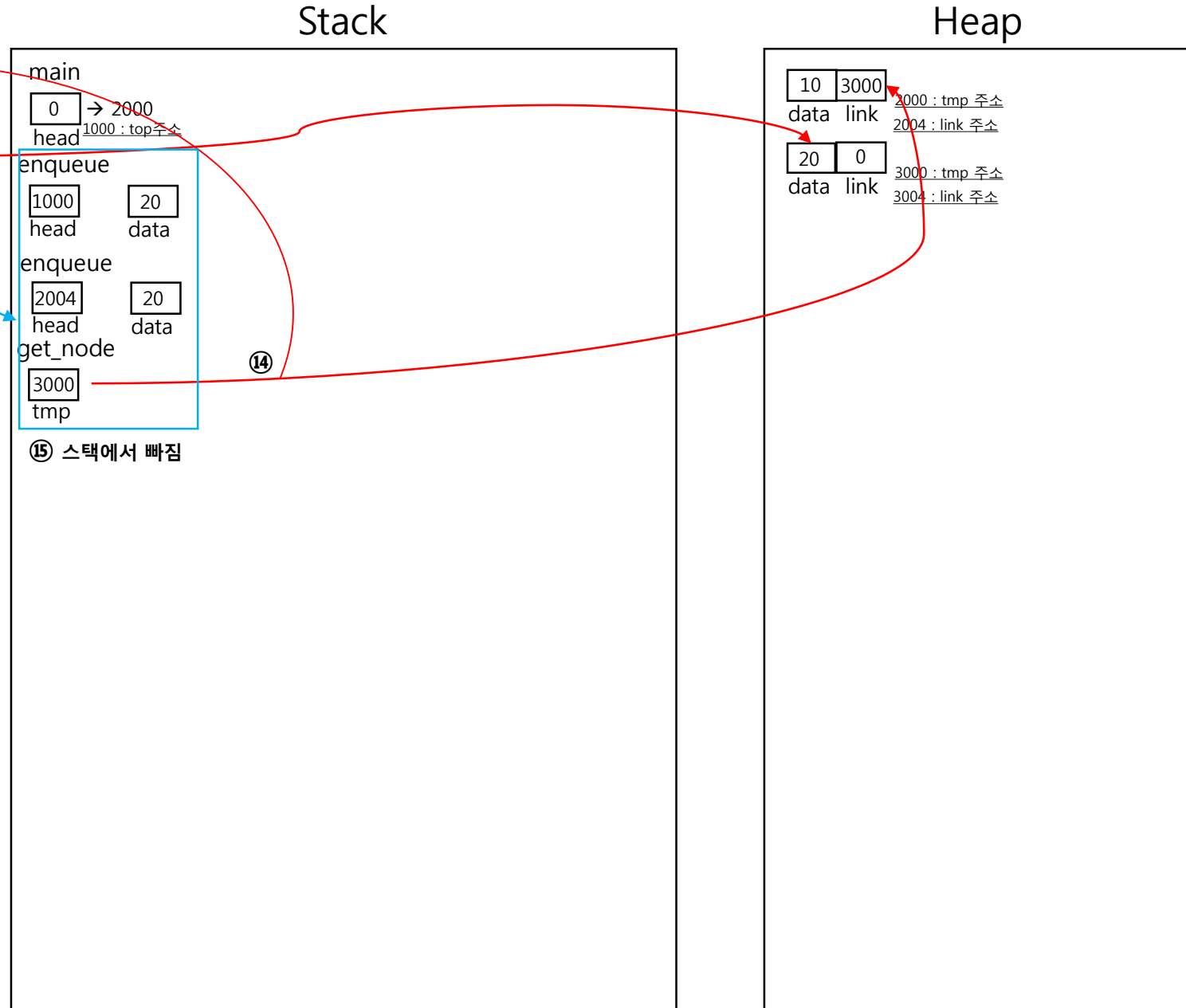
    enqueue(&head, 10);
    enqueue(&head, 20);
    enqueue(&head, 30);

    print_queue(head);

    dequeue(&head,20);

    print_queue(&head);

    return 0;
}
```



1.큐(Queue) 코드 이해(그림 그리기) - (enqueue 5)

Stack

Heap

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

```
#define EMPTY 0
```

```
struct node{
    int data;
    struct node *link;
};
```

```
typedef struct node Queue;
```

```
Queue *getnode(){
    Queue *tmp;
    tmp = (Queue *)malloc(sizeof(Queue));
    tmp->link = EMPTY;
    return tmp;
}
```

```
void dequeue(Queue **head, int data ){
```

```
    Queue *tmp;
    if( *head == EMPTY){
        printf("QUEUE IS EMPTY\n");
    }
```

```
    tmp = *head;
    if(tmp->data ==data){
```

```
        *head = tmp->link;
        free(tmp);
    }else
        dequeue( &(amp;tmp->link), data);
    }
```

```
void enqueue(Queue **head, int data){
    if( *head == EMPTY){
```

```
        *head = getnode();
        (*head)->data = data;
        return;
    }
    enqueue( &((*head)->link), data);
}
```

```
void print_queue(Queue *head){
```

```
    Queue *tmp = head;
```

```
    while(tmp){
        printf("%d\n",tmp->data);
        tmp = tmp->link;
    }
```

```
}
```

```
int main(void){
```

```
    Queue *head = EMPTY;
```

```
    enqueue(&head, 10);
    enqueue(&head, 20);
    enqueue(&head, 30);
```

```
    print_queue(head);
```

```
    dequeue(&head,20);
```

```
    print_queue(&head);
```

```
    return 0;
}
```

main

0 → 2000
head → 1000 : top주소

① ~ ⑮와 같은 방식으로 반복

10 3000
data link
2000 : tmp 주소
2004 : link 주소

20 4000
data link
3000 : tmp 주소
3004 : link 주소

30 0
data link
4000 : tmp 주소
4004 : link 주소

1. 큐(Queue) 코드 이해(그림 그리기) - (dequeue 1)

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

```
#define EMPTY 0
```

```
struct node{
    int data;
    struct node *link;
};
```

```
typedef struct node Queue;
```

```
Queue *getnode(){
    Queue *tmp;
    tmp = (Queue *)malloc(sizeof(Queue));
    tmp->link = EMPTY;
    return tmp;
}
```

```
void dequeue(Queue **head, int data ){
```

```
    Queue *tmp;
    if( *head == EMPTY){
        printf("QUEUE IS EMPTY\n");
    }
```

```
    tmp = *head;
    if(tmp->data == data){
        *head = tmp->link;
        free(tmp);
    }else
        dequeue( &(tmp->link), data);
}
```

```
void enqueue(Queue **head, int data){
    if( *head == EMPTY){
        *head = getnode();
        (*head)->data = data;
        return;
    }
    enqueue( &((*head)->link), data);
}
```

```
void print_queue(Queue *head){
    Queue *tmp = head;
    while(tmp){
        printf("%d\n",tmp->data);
        tmp = tmp->link;
    }
}
```

```
int main(void){
    Queue *head = EMPTY;

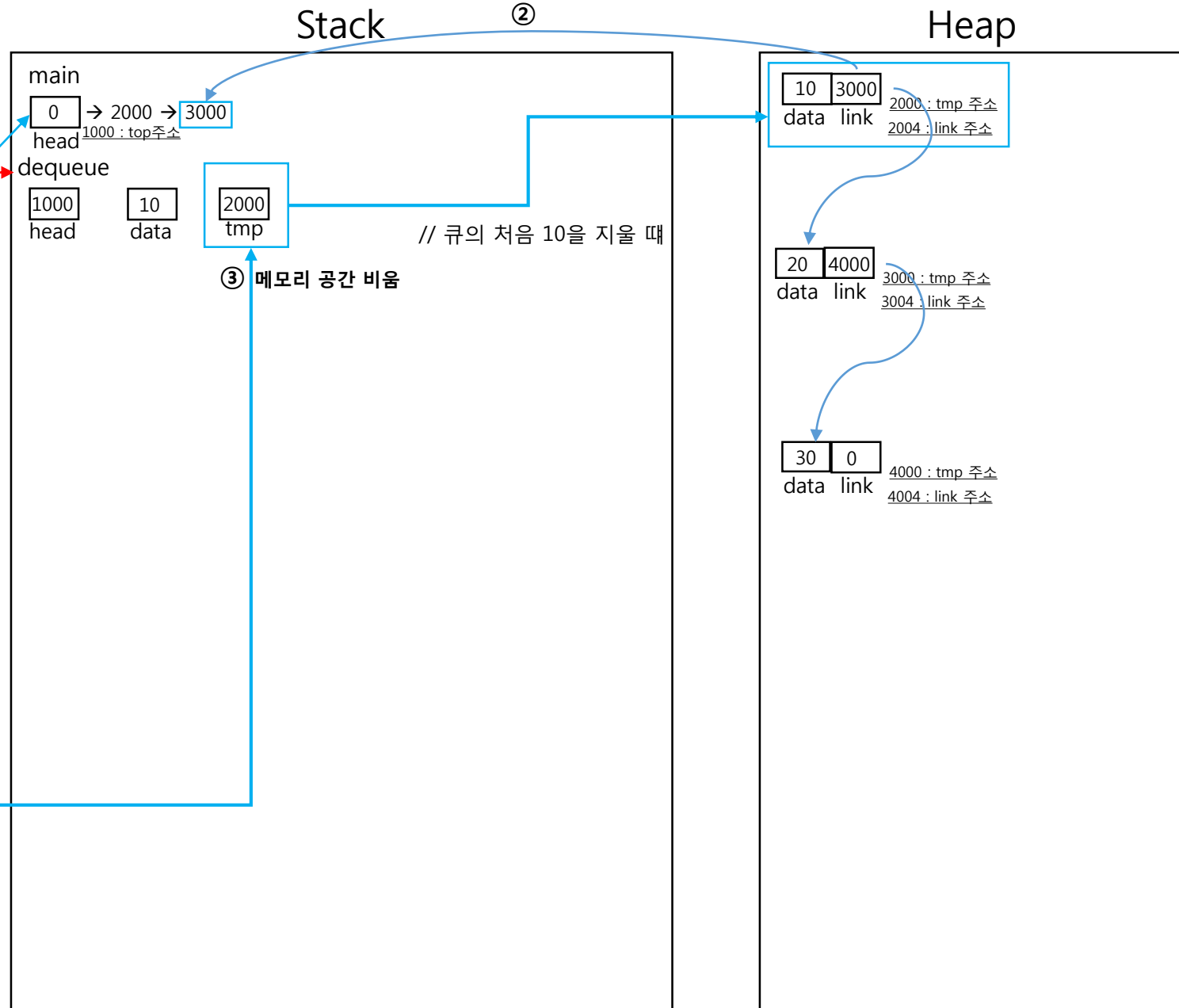
    enqueue(&head, 10);
    enqueue(&head, 20);
    enqueue(&head, 30);

    print_queue(head);

    dequeue(&head,10);

    print_queue(&head);

    return 0;
}
```



1.큐(Queue) 코드 이해(그림 그리기) - (dequeue 2)

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

```
#define EMPTY 0
```

```
struct node{
```

```
int data;
struct node *link;
};
```

```
typedef struct node Queue;
```

```
Queue *getnode(){
Queue *tmp;
tmp = (Queue *)malloc(sizeof(Queue));
tmp->link = EMPTY;
return tmp;
}
```

```
void dequeue(Queue **head, int data ){
```

```
Queue *tmp;
if( *head == EMPTY){

printf("QUEUE IS EMPTY\n");
}
```

```
tmp = *head;
if(tmp->data == data){
```

```
*head = tmp->link;
free(tmp);
}else
dequeue( &(amp;tmp->link), data);
}
```

```
void enqueue(Queue **head, int data){
if( *head == EMPTY){
```

```
*head = getnode();
(*head)->data = data;
return;
}
enqueue( &((*head)->link), data);
}
```

```
void print_queue(Queue *head){
```

```
Queue *tmp = head;
```

```
while(tmp){

printf("%d\n",tmp->data);
tmp = tmp->link;
}
```

```
}
```

```
int main(void){
```

```
Queue *head = EMPTY;
```

```
enqueue(&head, 10);
enqueue(&head, 20);
enqueue(&head, 30);
```

```
print_queue(head);
```

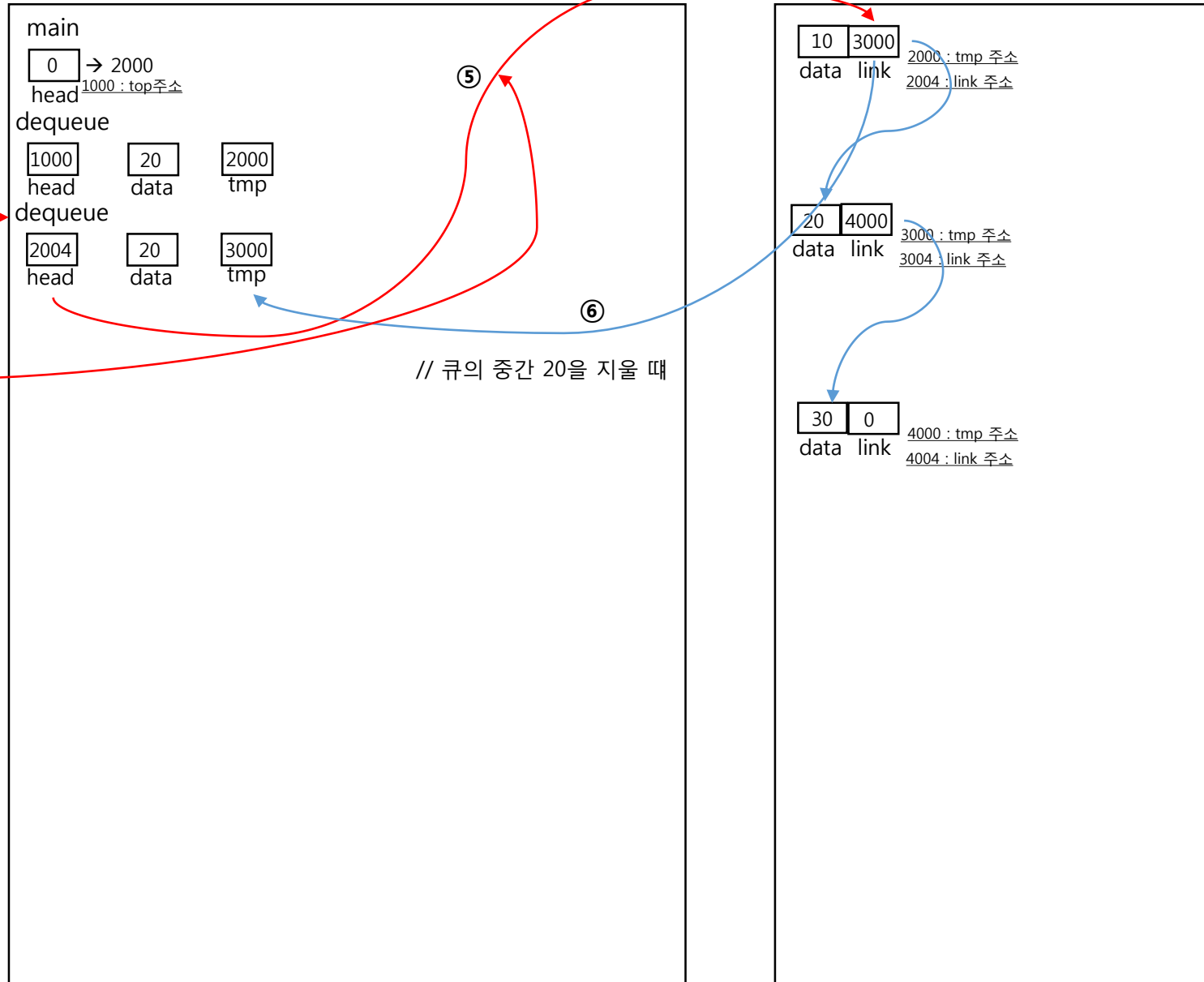
```
dequeue(&head,20);
```

```
print_queue(&head);
```

```
return 0;
}
```

Stack

Heap



1.큐(Queue) 코드 이해(그림 그리기) - (dequeue 3)

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

```
#define EMPTY 0
```

```
struct node{
```

```
int data;
struct node *link;
};
```

```
typedef struct node Queue;
```

```
Queue *getnode(){
Queue *tmp;
tmp = (Queue *)malloc(sizeof(Queue));
tmp->link = EMPTY;
return tmp;
}
```

```
void dequeue(Queue **head, int data ){
```

```
Queue *tmp;
if( *head == EMPTY){

printf("QUEUE IS EMPTY\n");
}
```

```
tmp = *head;
if(tmp->data ==data){
```

```
*head = tmp->link;
```

```
free(tmp);
```

```
}else
dequeue( &(amp;tmp->link), data);
}
```

```
void enqueue(Queue **head, int data){
if( *head == EMPTY){
```

```
*head = getnode();
(*head)->data = data;
return;
}
enqueue( &((*head)->link), data);
}
```

```
void print_queue(Queue *head){
```

```
Queue *tmp = head;
```

```
while(tmp){

printf("%d\n",tmp->data);
tmp = tmp->link;
}
```

```
}
```

```
int main(void){
```

```
Queue *head = EMPTY;
```

```
enqueue(&head, 10);
enqueue(&head, 20);
enqueue(&head, 30);
```

```
print_queue(head);
```

```
dequeue(&head,20);
```

```
print_queue(&head);
```

```
return 0;
```

```
}
```

main

0 → 2000
head 1000 : top주소

dequeue

1000 20 2000
head data tmp

dequeue

2004 20 3000
head data tmp

Stack

⑧ 메모리 공간 비움

// 큐의 중간 20을 지울 때

⑦

Heap

10 4000
data link
2000 : tmp 주소
2004 : link 주소

20 4000
data link
3000 : tmp 주소
3004 : link 주소

30 0
data link
4000 : tmp 주소
4004 : link 주소

1.큐(Queue) 코드 이해(그림 그리기) - (dequeue 4)

Stack

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

```
#define EMPTY 0
```

```
struct node{
```

```
int data;
struct node *link;
};
```

```
typedef struct node Queue;
```

```
Queue *getnode(){
Queue *tmp;
tmp = (Queue *)malloc(sizeof(Queue));
tmp->link = EMPTY;
return tmp;
}
```

```
void dequeue(Queue **head, int data ){
```

```
Queue *tmp;
if( *head == EMPTY){

printf("QUEUE IS EMPTY\n");
}
```

```
tmp = *head;
if(tmp->data ==data){
```

```
*head = tmp->link;
free(tmp);
}else
dequeue( &(amp;tmp->link), data);
}
```

```
void enqueue(Queue **head, int data){
if( *head == EMPTY){
```

```
*head = getnode();
(*head)->data = data;
return;
}
enqueue( &((*head)->link), data);
}
```

```
void print_queue(Queue *head){
```

```
Queue *tmp = head;
```

```
while(tmp){

printf("%d\n",tmp->data);
tmp = tmp->link;
}
```

```
}
```

```
int main(void){
```

```
Queue *head = EMPTY;
```

```
enqueue(&head, 10);
enqueue(&head, 20);
enqueue(&head, 30);
```

```
print_queue(head);
```

```
dequeue(&head,20);
```

```
print_queue(&head);
```

```
return 0;
}
```

main

0 → 2000
head 1000 : top주소

dequeue

1000 20 2000
head data tmp

dequeue

2004 20 3000
head data tmp

⑨ 스택에서 빠짐

// 큐의 중간 20을 지울 때

Heap

10 4000
data link 2000 : tmp 주소
2004 : link 주소

30 0
data link 4000 : tmp 주소
4004 : link 주소