Xilinx Zynq FPGA,TI DSP, MCU 기반의 프로그래밍 전문가 과정

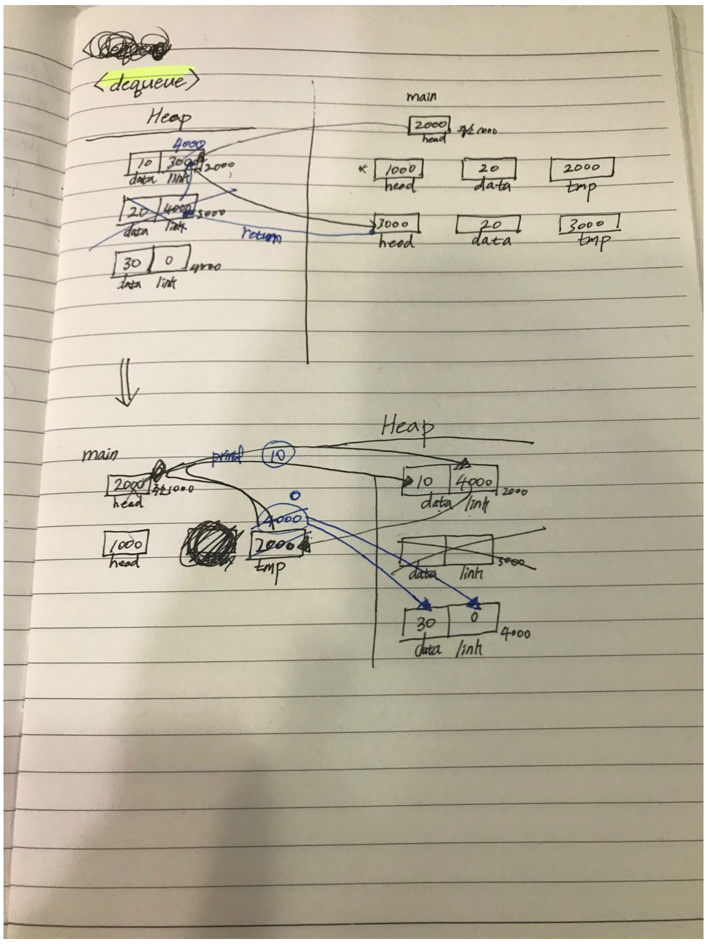
강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 정한별
hanbulkr@gmail.com

<dequeue>

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

#define EMPTY 0

typedef struct __queue
{
	int data;
	struct __queue *link;

}queue;

queue *get_node(){

	queue *tmp;
	tmp = (queue *)malloc(sizeof(queue));
	tmp -> link = EMPTY;

	return tmp;
}

void enqueue(queue **head, int data){

	if(*head == NULL){
		*head = get_node();
		(*head) -> data = data;

		return ;
	}

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

}
void queue_delete(queue *head,int data)
{
	queue *tmp;
```

```
		tmp = head;
		while(tmp)
		{
				if((tmp -> data) == data){
				//		printf("같습니다.%d\n",data);
						tmp = tmp ->link;

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

void queue_delete2(queue *head, int data)
{
		queue *tmp;
		tmp = head;
		if((tmp->data) == data)
		{
				head -> link = tmp -> link;
				printf("같습니다.\n");
				free(tmp);
		}
		else if((tmp->data) != data)
		{
				head->link = tmp -> link;
				printf("res = %d\n", tmp -> data);
		}
		else
						return ;

		queue_delete2( (tmp->link) , data);
}

queue *queue_delete3(queue *head, int data)
{
		queue *tmp = head;

		if(tmp ==NULL)
						printf("There are no data that you delete\n");
		if(head ->data != data)
						head ->link = queue_delete3(head->link, data);
		else
		{
						// queue *res = head ->link;
						printf("Now you delete %d\n",data);
```

```
                        free(tmp);
                        return head->link;
            }
            return head;
}

int main(void){

            queue *heap = EMPTY;
//          srand(time(NULL));
            enqueue(&heap, 10);
            enqueue(&heap, 20);
            enqueue(&heap, 30);
            print_queue(heap);
            queue_delete3(heap,20);
//          heap = queue_delete3(heap,20);
//          queue_delete2(heap,20);
//          print_queue(heap);
//          queue_delete(heap,20);

            print_queue(heap);
            return 0;
}
```



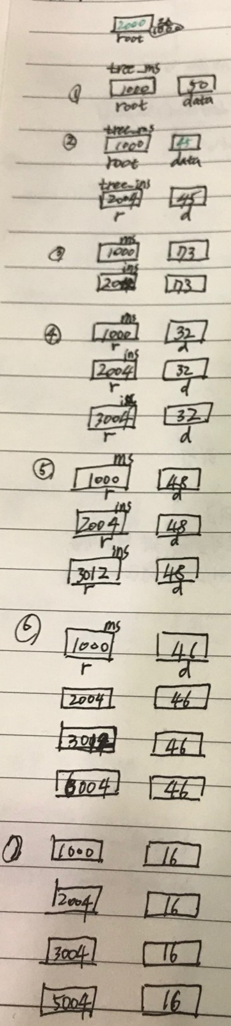
<이진트리>

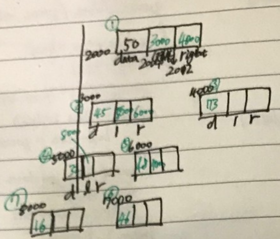
```
#include<stdio.h>
#include<malloc.h>
#include<stdlib.h>
#include<time.h>

#define EMPTY 0

typedef struct __tree
{
		int data;
//		int data2;
		struct __tree *link_right;
		struct __tree *link_left;

}tree;

tree *get_node()
{
		tree *tmp;
		tmp = (tree *)malloc(sizeof(tree));
		tmp -> link_right = EMPTY;
		tmp -> link_left = EMPTY;
		return tmp;
}



void binary(tree **root, int data)
{
		tree *tmp = *root;
		if( *root == NULL){
				*root = get_node();
				(*root) -> data =data;
				return ;
		}

		else if((*root)->data > data){
		//		(*root)->link_left = tmp;
				binary(&((*root)->link_left), data);
		}
		else if((*root)->data <data){
		//		(*root)->link_right = tmp;
				 binary(&((*root)->link_right), data);
		}
		else
				printf("값이 같습니다.\n");
```

```
}

void print(tree *top)
{
        if(top)
        {
                printf("data = %d, ", top -> data);

                if(top->link_left)
                                printf("left = %d, " , top ->link_left->data);
                else
                                printf("left = NULL, ");
                if (top ->link_right)
                                printf("right =%d \n", top ->link_right->data);
                else
                                printf("right = NULL\n");



                print(top->link_left);
                // 맨위에 print(tree *top) 속에 top 에 뭘 던질 것이냐...
                print(top->link_right);
                // 맨위에 print(
        }

}

int main(void)
{
        int a[] = {50,45,73,32,48,46,16,37,120,47,130,127,124};
        int len = (sizeof(a)/sizeof(int));
        int i;
        srand((unsigned)time(NULL));

        tree *top = EMPTY;
        for(i=0;i<len;i++)
                binary( &top, a[i]);

        printf("%d\n",len);
        print(top);


//      a[rand()%(sizeof(a)/(int))]


        return 0;
}
```

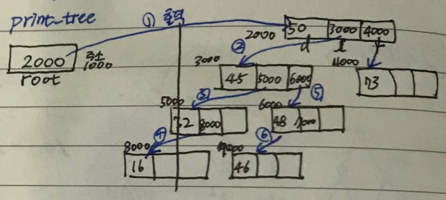
<binary_tree>

main





<print>



* 맨 왼쪽부터 차례로 출력후
다시 올라오면서 node의 오른쪽이 (가지) 있을 시
~~[illegible]~~ 오른 node의 왼쪽부터 실행한다.
얼마전 피보나치 처럼 실행됨.