TI DSP,MCU및 Xilinux Zynq FPGA 프로그래밍 전문가 과정

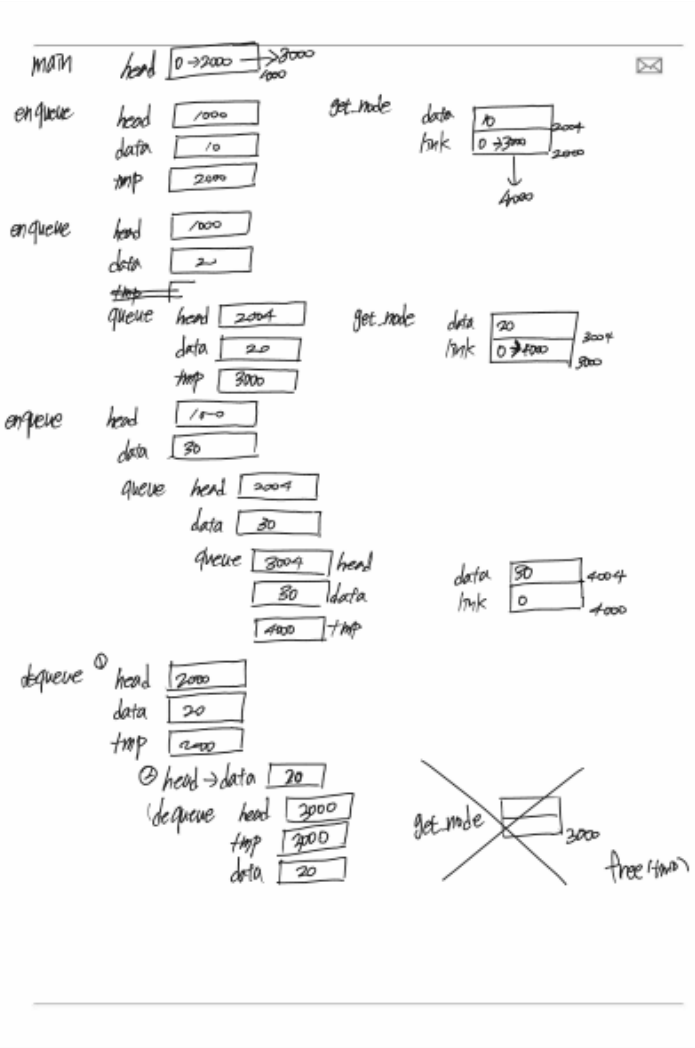
이름	문지희
학생 이메일	mjh8127@naver.com
날짜	2018/3/6
수업일수	9일차
담당강사	Innova Lee(이상훈)
강사 이메일	gcccompil3r@gmail.com

목차

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

#define EMPTY 0

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

typedef struct node 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);
        printf("test₩n");
}
```

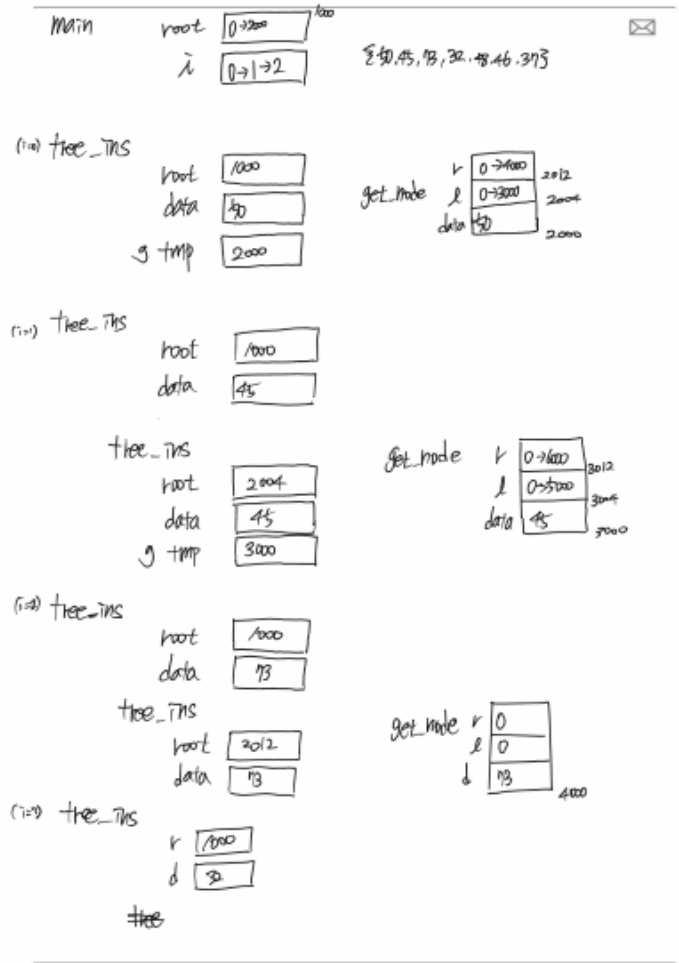
```
void print_queue(queue*head)
{
        queue *tmp = head;
        while(tmp)
        {
                printf("%d₩n", tmp -> data);
                tmp=tmp->link;
        }
        return ;
}
queue *dequeue(queue *head, int data)
{
        queue *tmp = head;

        if(tmp == NULL)
                printf("There are no data that you delete₩n");

        if(head->data != data)
                head->link = dequeue(head->link, data);

        else
        {
                //queue *res = head -> link;
                printf("Now you delete %d₩n", data);
                free(tmp);
                return head -> link;
        }
        return head;
}
```

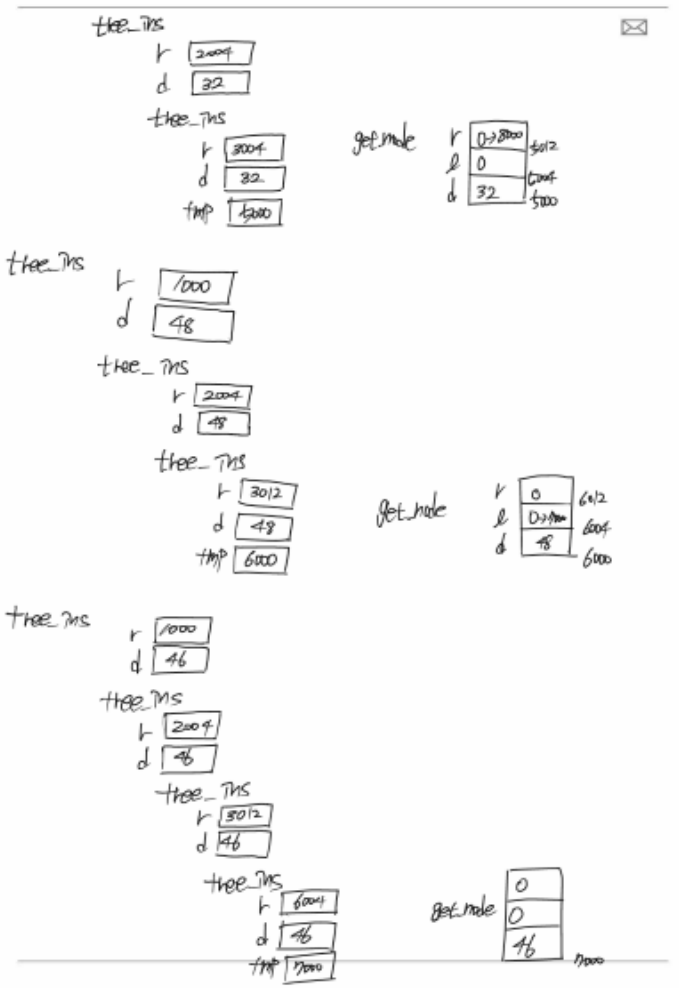
```
int main (void)
{
        int i;

        queue *head = NULL;

        srand(time(NULL));
        for(i=0; i<3; i++)
                enqueue(&head, (i+1)*10);

        print_queue(head);
        head = dequeue(head, 20);
        print_queue(head);
        return 0;
}
```

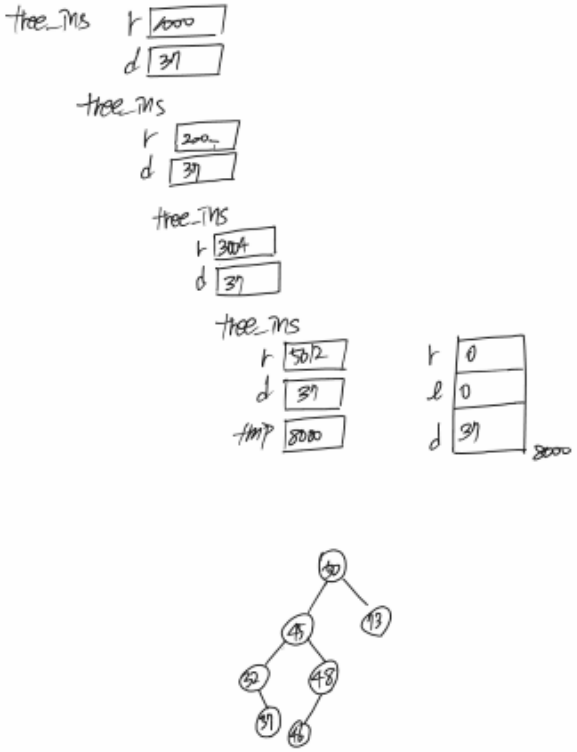

main
head
0 → 2000
3000
1000
enqueue
head
1000
data
10
tmp
2000
get_node
data
10
2004
link
0 → 3000
2000
4000
enqueue
head
1000
data
20
tmp
queue
head
2004
data
20
tmp
3000
get_node
data
20
3004
link
0 → 4000
3000
enqueue
head
1000
data
30
queue
head
3004
data
30
queue
3004
head
30
data
4000
tmp
data
50
4004
link
0
4000
dequeue ①
head
2000
data
20
tmp
2000
② head → data
20
(dequeue
head
3000
tmp
3000
data
20
get_node
3000
free (tmp)

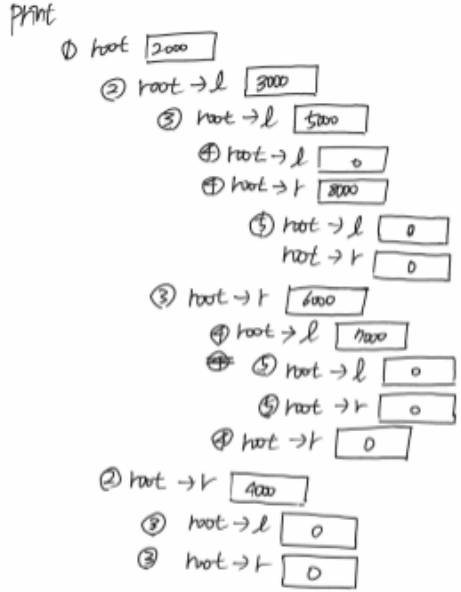
```
#include<stdio.h>
#include<malloc.h>

typedef struct __tree{
        struct __tree *l;
        struct __tree *r;
        int data;
}tree;


tree *get_node()
{
        tree *tmp;
        tmp=(tree*)malloc(sizeof(tree));
        tmp->r=NULL;
        tmp->l=NULL;
        return tmp;
}

void division(tree **head,int data)
{
        if(*head==NULL)
        {
        *head=get_node();
        (*head)->data=data;
        return ;
        }

        else if((*head)->data < data)
        {
                division(&(*head)->r,data);
        }
        else if((*head)->data > data)
        {
                division(&(*head)->l,data);
        }


}

void search(tree *head,int input)
   {
        if(head->data==input)
                printf("찾음₩n");

        else if((head->data) > input)
        {
                printf("다음₩n");
                search(head->l,input);
        }
        else if((head->data) < input)
        {
                printf("다음₩n");
                search(head->r,input);
        }


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

                if(head->l)
                        printf("left=%d",head->l->data);
                else
                        printf("left=NULL,");
```

```
                if(head->r)
                        printf("right=%d₩n",head->r->data);
                else
                        printf("right=NULL₩n");

                print(head->l);
                print(head->r);

        }
}



void delete(tree *head ,int d)
{
        tree *pre=head->;
        if(head->data==d)
        {
        free(tmp);

        }
        else if((head->data) > d)
        {

                delete(head->l,d);
        }
        else if((head->data) < d)
        {

                delete(head->r,d);
        }
}
int main(void)
{
        tree *head=NULL;
        int i,data,/*size,*/input,d;
        int arr[]={50, 45, 73, 32, 48, 46, 16, 37, 120, 47, 130,
127, 124};
        //size=sizeof(arr)/sizeof(int);
        //end=arr[size];

        for(i=0;arr[i] /*i<size*/;i++)
          {
                //data=arr[i];
                division(&head,arr[i]);
          }
        printf("_____₩n");
        //print(head);
        input=124;
        d=47;
        search(head,input);
        delete(head,d);
        return 0;
```


main
root
0→1000
i
0→1→2
{50,45,13,32,48,46,375}
(1:0) tree_ins
root
1000
data
50
tmp
2000
get_node
r 0→1000 2012
l 0→3000 2004
data 50 2000
(1:1) tree_ins
root
1000
data
45
tree_ins
root
2004
data
45
tmp
3000
get_node
r 0→1000 3012
l 0→5000 3004
data 45 3000
(1:2) tree_ins
root
1000
data
13
tree_ins
root
3012
data
13
get_node
r 0
l 0
d 13 4000
(1:3) tree_ins
r 1000
d 32
tree


tree_ins
r 2004
d 32
tree_ins
r 3004
d 32
tmp 5000
get_node
r 0→3000 5012
l 0 6004
d 32 5000
tree_ins
r 1000
d 48
tree_ins
r 2004
d 48
tree_ins
r 3012
d 48
tmp 6000
get_node
r 0 6012
l 0→5000 6004
d 48 6000
tree_ins
r 1000
d 46
tree_ins
r 2004
d 46
tree_ins
r 5012
d 46
tree_ins
r 6004
d 46
tmp 7000
get_node
0
0
46 7000


tree_ins
r 1000
d 37
tree_ins
r 2000
d 37
tree_ins
r 3004
d 37
tree_ins
r 5012
d 37
tmp 8080
r 0
l 0
d 37
8000
50
45
13
42
48
37
46


print
① root 2000
② root→l 3000
③ root→l 5000
④ root→l 0
④ root→r 8000
⑤ root→l 0
root→r 0
③ root→r 6000
④ root→l 7000
⑤ root→l 0
⑤ root→r 0
④ root→r 0
② root→r 4000
③ root→l 0
③ root→r 0