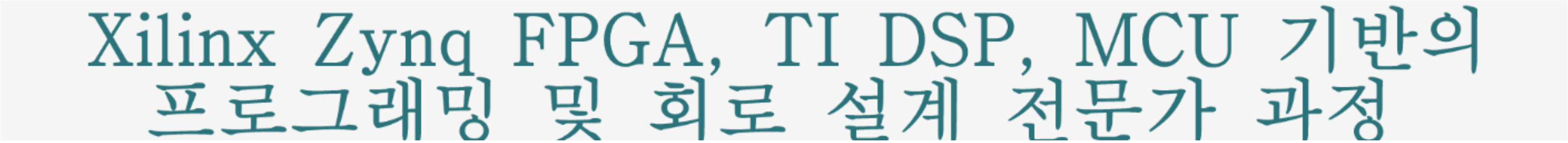
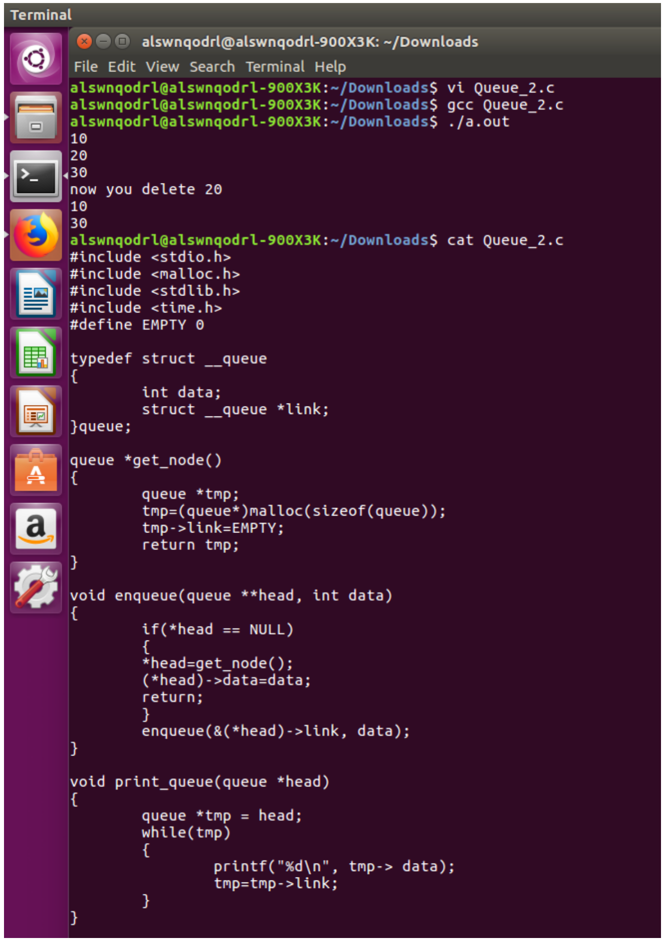


Xilinx Zynq FPGA, TI DSP, MCU 기반의 프로그래밍 및 회로 설계 전문가 과정

강사 – Innova Lee (이상훈)
gcccompil3r@gmail.com
학생-김민주
alswnqodrl@naver.com

01

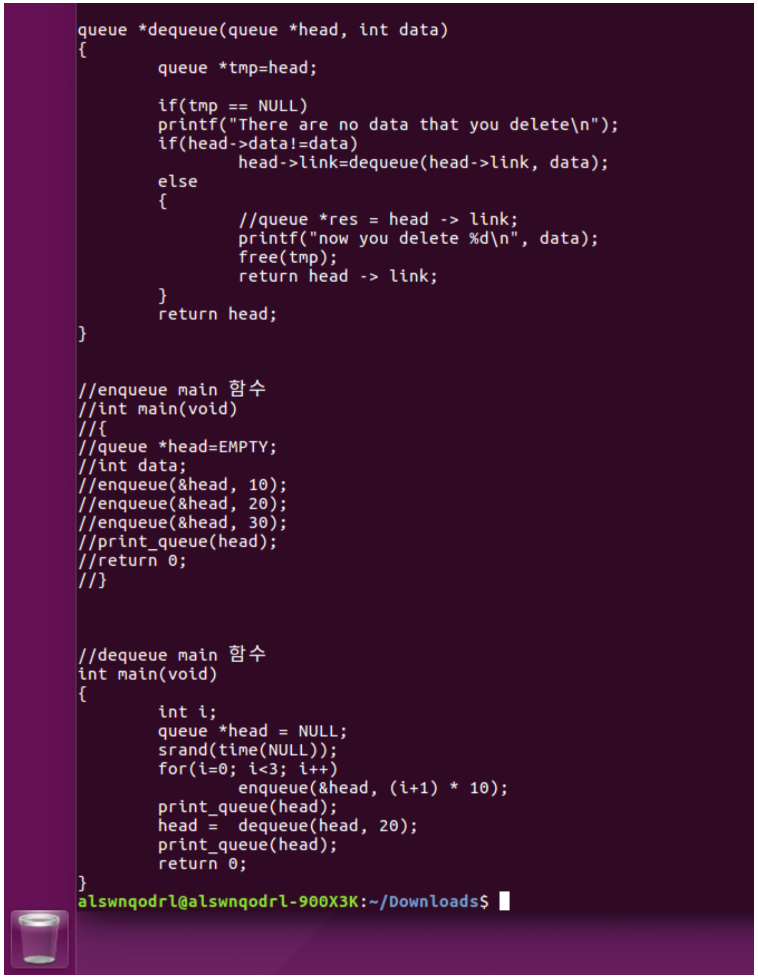
9일차 내용 복습
(enqueue, dequeue)


Terminal
alswnqodrl@alswnqodrl-900X3K: ~/Downloads
File Edit View Search Terminal Help

```
alswnqodrl@alswnqodrl-900X3K:~/Downloads$ vi Queue_2.c
alswnqodrl@alswnqodrl-900X3K:~/Downloads$ gcc Queue_2.c
alswnqodrl@alswnqodrl-900X3K:~/Downloads$ ./a.out
10
20
30
now you delete 20
10
30
alswnqodrl@alswnqodrl-900X3K:~/Downloads$ cat Queue_2.c
#include <stdio.h>
#include <malloc.h>
#include <stdlib.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 = head;
        while(tmp)
        {
                printf("%d\n", tmp-> data);
                tmp=tmp->link;
        }
}
```

01

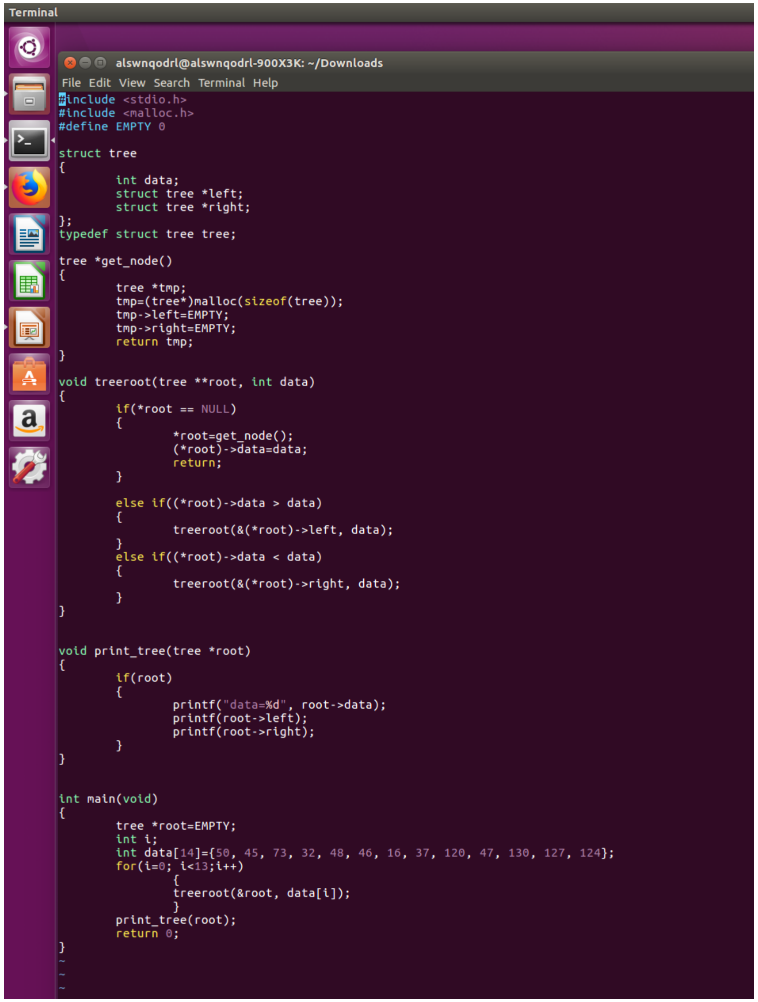
9일차 내용 복습
(enqueue, dequeue)



```
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;
}


//enqueue main 함수
//int main(void)
//{
//queue *head=EMPTY;
//int data;
//enqueue(&head, 10);
//enqueue(&head, 20);
//enqueue(&head, 30);
//print_queue(head);
//return 0;
//}



//dequeue main 함수
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;
}
alswnqodrl@alswnqodrl-900X3K:~/Downloads$ 
```

02

9일차 내용 복습(tree)


Terminal
alswnqodrl@alswnqodrl-900X3K: ~/Downloads
File Edit View Search Terminal Help

```
#include <stdio.h>
#include <malloc.h>
#define EMPTY 0

struct tree
{
        int data;
        struct tree *left;
        struct tree *right;
};
typedef struct tree tree;

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

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

        else if((*root)->data > data)
        {
                treeroot(&(*root)->left, data);
        }
        else if((*root)->data < data)
        {
                treeroot(&(*root)->right, data);
        }
}


void print_tree(tree *root)
{
        if(root)
        {
                printf("data=%d", root->data);
                printf(root->left);
                printf(root->right);
        }
}


int main(void)
{
        tree *root=EMPTY;
        int i;
        int data[14]={50, 45, 73, 32, 48, 46, 16, 37, 120, 47, 130, 127, 124};
        for(i=0; i<13;i++)
                {
                treeroot(&root, data[i]);
                }
        print_tree(root);
        return 0;
}
~
~
~
```

03

8일차 내용 복습 – 그림그리기
(enqueue)

Main 0
Head 1000->2000

enqueue head(1000)	data(10)
Enqueue head(2004)	data(20)
enqueue head(3004)	data(30)

Heap

10 data(2000)	0->3000 link
~~20~~ ~~data(3000)~~	~~4000~~ ~~link~~
30 data(4000)	0 link

04

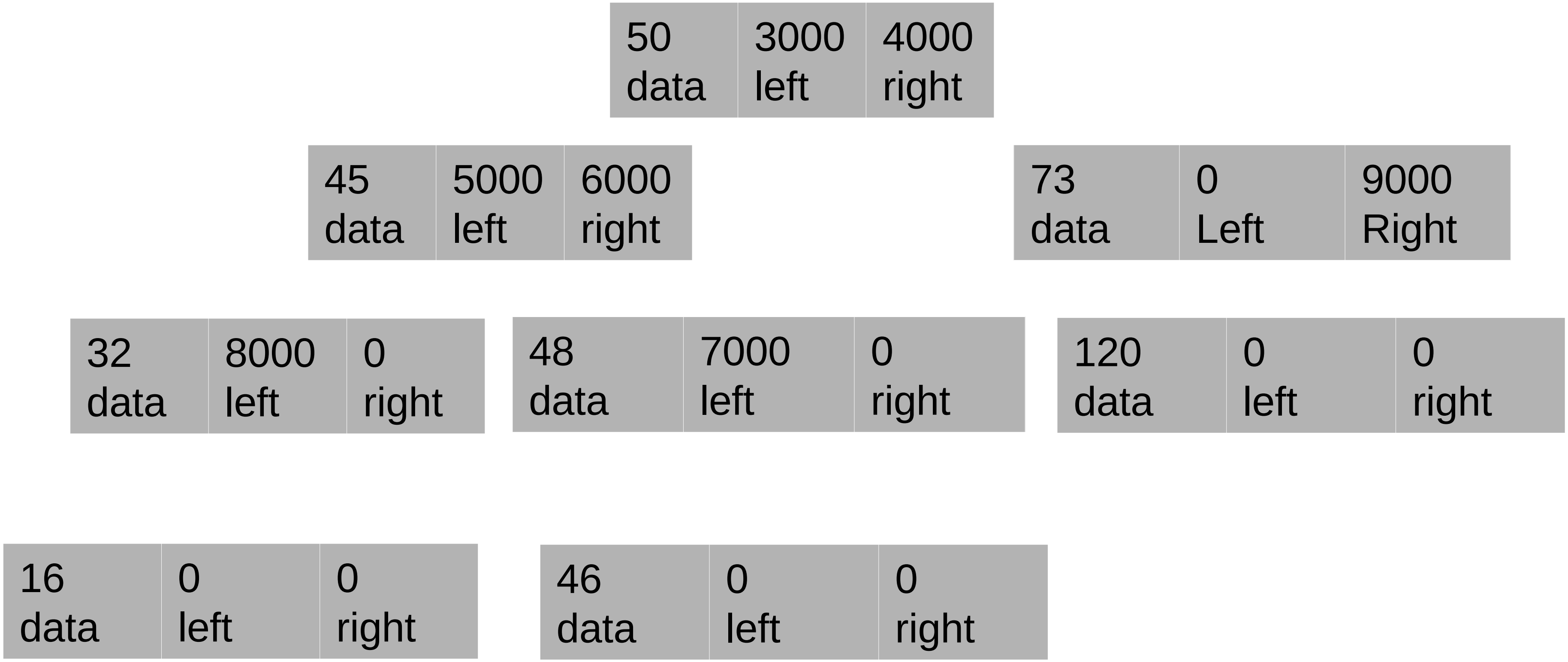
8일차 내용 복습 – 그림그리기 (dequeu)

2000 head	20 data	2000 tmp
2000 head	20 data	tmp

10 data	3000->4000 link
~~20~~ ~~data~~	~~4000~~ ~~=>return됨~~ ~~link~~
30 data	0 link

05

8일차 내용 복습 – 그림그리기 (tree)



50 data	3000 left	4000 right

45 data	5000 left	6000 right

73 data	0 Left	9000 Right

32 data	8000 left	0 right

48 data	7000 left	0 right

120 data	0 left	0 right

16 data	0 left	0 right

46 data	0 left	0 right