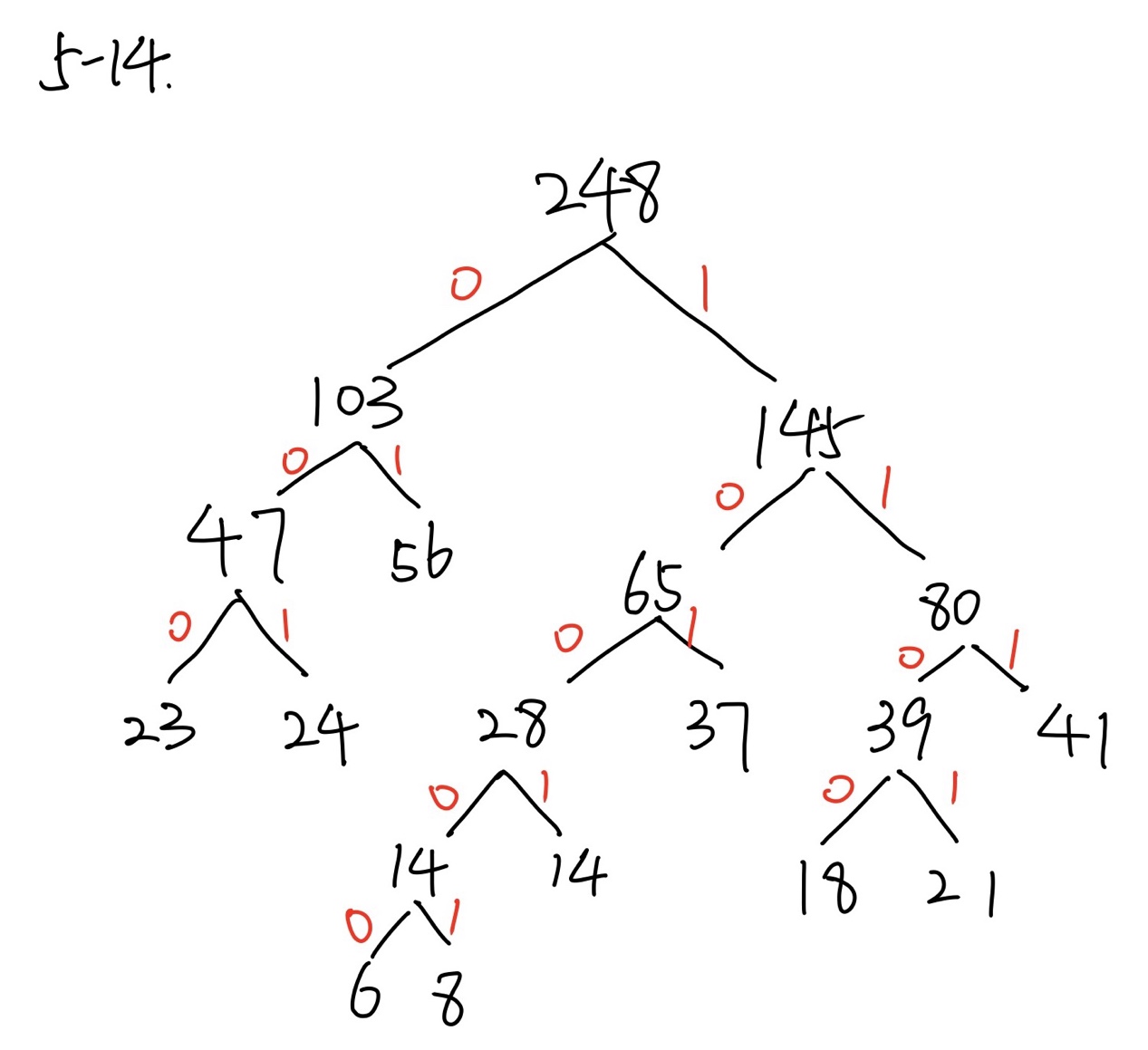
**数据结构习题**

**学号： 姓名：**

**1、**5-14

图中所给数据，构造最优二叉树。如下图



故可得编码：

8: 10001

21: 1101

37: 101

24: 001

6: 10000

18: 1100

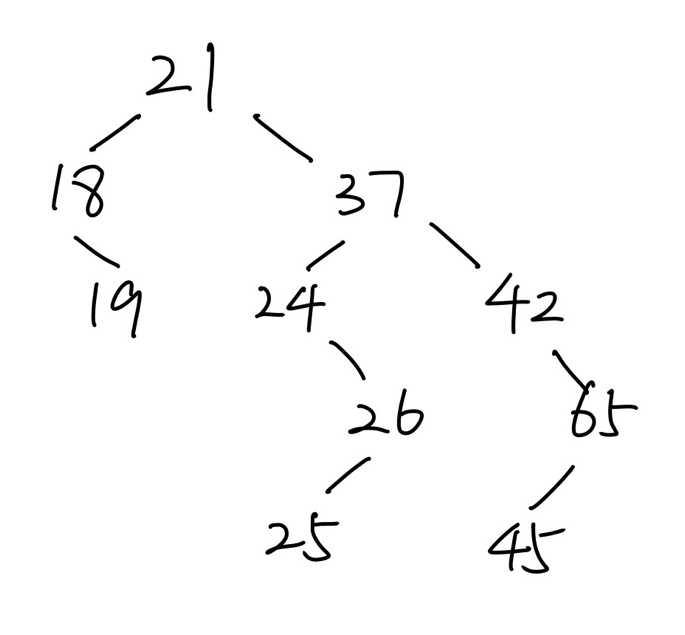
23: 000

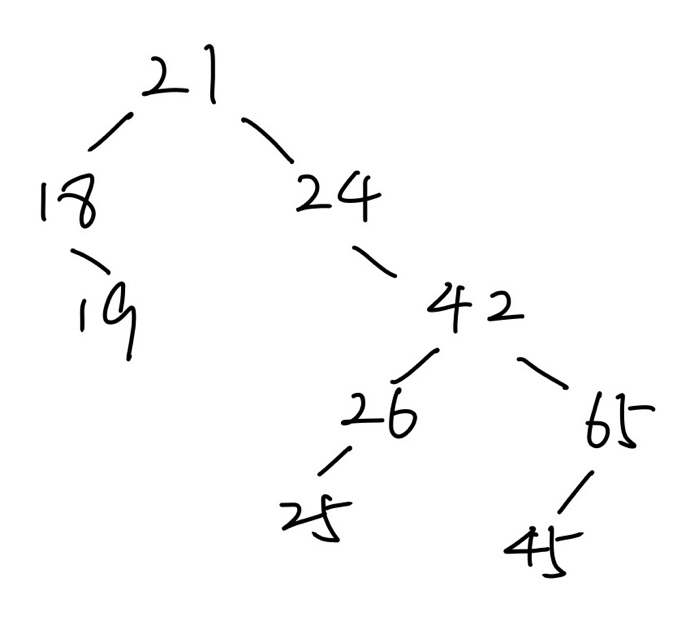
41: 111

56: 01

14: 1001

**2、**5-15

删除前：

删除后：

**3、**5-17

#include <stdio.h>

#include <stdlib.h>

const int MAXSIZE = 1024;

typedef struct node

{

char data;

struct node \*lchild, \*rchild;

}bitree;

bitree \*CreateTree()

{

char ch; // 接受输入变量

bitree \*Q[MAXSIZE]; // 定义存放树中元素的队列，最简单队列即可，方便树的结点的安排

int front = 1, rear = 0; // 队头和队尾

bitree \*root = NULL, \*s; // 定义树的根节点和当前节点

while((ch = getchar()) != '#') // 当输入不为#的时候进入循环，构造二叉树

{

s = NULL; // 将当前节点置空

if(ch != '@') // @代表结点元素为空，此时不为节点分配空间

{

s = (bitree \*)malloc(sizeof(bitree));

s -> data = ch;

s -> lchild = NULL;

s -> rchild = NULL;

}

Q[ ++ rear] = s; // 将当前节点放入队列

if(rear == 1) // 当前队尾为1时，说明头结点为空，将此节点作为头结点

root = s;

else

{

if(s && Q[front]) // 当前节点不为空（即元素不为@），并且队首节点（即当前节点的母节点）不为空，构建front结点的树的左右母子关系

{

if(rear % 2 == 0) // 当队尾结点（编号和树里编号一致），当队尾元素为偶数时，左子节点，队尾元素为奇数时，右子节点

Q[front] -> lchild = s;

else

Q[front] -> rchild = s;

}

if(rear % 2 == 1) // 当队列尾结点下标为奇数，说明节点Q【front】的两个孩子节点处理完毕，出队

front ++ ;

}

}

return root; // 返回树根节点

}

void PreOrder(bitree \*root)

{

bitree \*stack[MAXSIZE];

int top = -1;

bitree \*p;

stack[ ++ top] = root;

while(root || top != -1)

{

p = stack[top -- ];

printf("%c ", p -> data);

if(p -> rchild)

stack[ ++ top] = p -> rchild;

if(p -> lchild)

stack[ ++ top] = p -> lchild;

}

}

int main()

{

printf("\n请输入二叉树中元素:\n");

bitree \*root = CreateTree();

printf("\n先序遍历的元素为:\n");

PreOrder(root);

printf("\n");

return 0;

}

**4、**5-21

#include <stdio.h>

#include <stdlib.h>

const int MAXSIZE = 1024;

typedef struct node

{

char data;

struct node \*lchild, \*rchild;

}bitree;

bitree \*CreateTree()

{

char ch; // 接受输入变量

bitree \*Q[MAXSIZE]; // 定义存放树中元素的队列，最简单队列即可，方便树的结点的安排

int front = 1, rear = 0; // 队头和队尾

bitree \*root = NULL, \*s; // 定义树的根节点和当前节点

while((ch = getchar()) != '#') // 当输入不为#的时候进入循环，构造二叉树

{

s = NULL; // 将当前节点置空

if(ch != '@') // @代表结点元素为空，此时不为节点分配空间

{

s = (bitree \*)malloc(sizeof(bitree));

s -> data = ch;

s -> lchild = NULL;

s -> rchild = NULL;

}

Q[ ++ rear] = s; // 将当前节点放入队列

if(rear == 1) // 当前队尾为1时，说明头结点为空，将此节点作为头结点

root = s;

else

{

if(s && Q[front]) // 当前节点不为空（即元素不为@），并且队首节点（即当前节点的母节点）不为空，构建front结点的树的左右母子关系

{

if(rear % 2 == 0) // 当队尾结点（编号和树里编号一致），当队尾元素为偶数时，左子节点，队尾元素为奇数时，右子节点

Q[front] -> lchild = s;

else

Q[front] -> rchild = s;

}

if(rear % 2 == 1) // 当队列尾结点下标为奇数，说明节点Q【front】的两个孩子节点处理完毕，出队

front ++ ;

}

}

return root; // 返回树根节点

}

void ChangeChild(bitree \*root)

{

bitree \*p;

if(root)

{

p = root -> lchild;

root -> lchild = root -> rchild;

root -> rchild = p;

ChangeChild(root -> lchild);

ChangeChild(root -> rchild);

}

}

int main()

{

bitree \*root = CreateTree();

ChangeChild(root);

return 0;

}