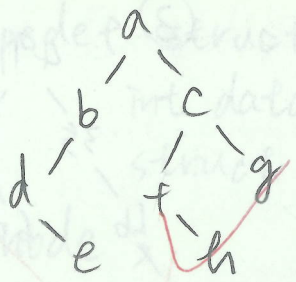
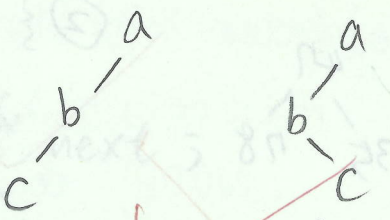
1. (a)



(b)



preorder : abc

postorder : cba

(c) $n_0 = 10$

$b = n-1 \Rightarrow 2n_2 = n_2 + n_0 - 1$

$\Rightarrow n_2$ (internal nodes) $= 9$

(d) $n+1$

(e) stack.

level-order traversal.

(f) inorder

由左至右的遞增數列

2.

insert 47 :

i	1	2	3	...
A[i]	47			

insert 35 :

i	1	2
A[i]	35	47

insert 78 :

i	1	2	3
A[i]	35	47	78

insert 14 :

1	2	3	4
14	35	78	47

delete min :

1	2	3
35	47	78

insert 51 :

i	1	2	3	4
	35	47	78	51

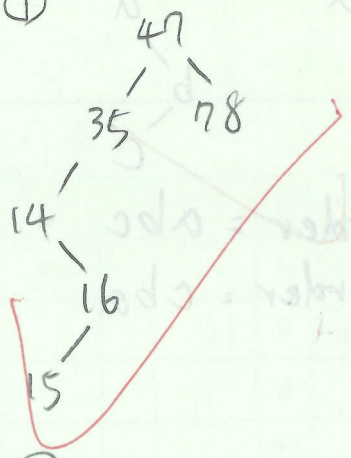
insert 3 :

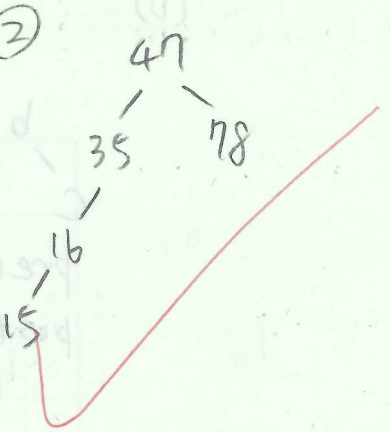
i	1	2	3	4	5
	3	35	78	51	47

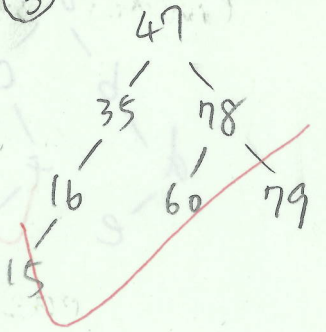
delete min :

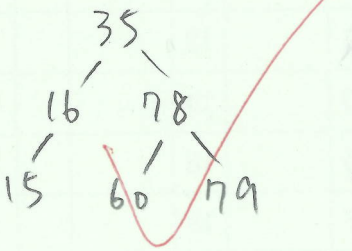
i	1	2	3	4
A[i]	35	47	78	51

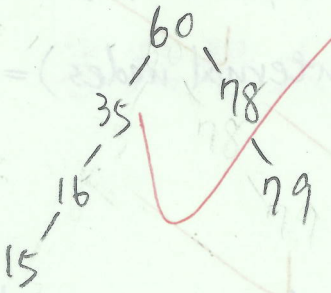
3. 10

① 

② 

③ 

④ 

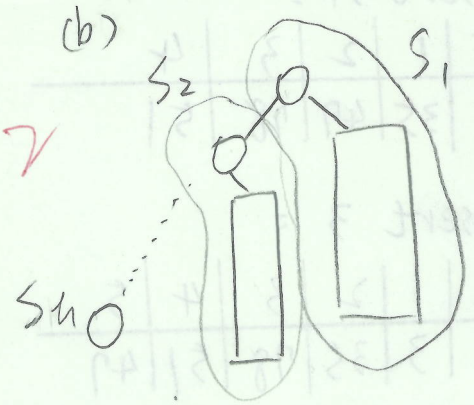
or 

4. (a) 如果 2 顆樹 node 數量 $|S_A|$ $|S_B|$ 相同：

union (A,B) : A = A∪B

否則 數量較少的集合 併入 node 數量較多的集合

node

(b) 

$m = S_1 + S_2 + S_3 + \cdots + S_h$

For Weighting Union :

$\sum_{n=k}^{h} |S_n| \le \frac{m}{2^{k-1}}$ for $1 \le k \le h$

Let $S_h = 1 \le \frac{m}{2^{h-1}}$

$\Rightarrow 2^{h-1} \le m \Rightarrow h \le \log_2 m + 1$

$\Rightarrow$ Height $= \lfloor h \rfloor = \lfloor \log_2 m \rfloor + 1$

5.

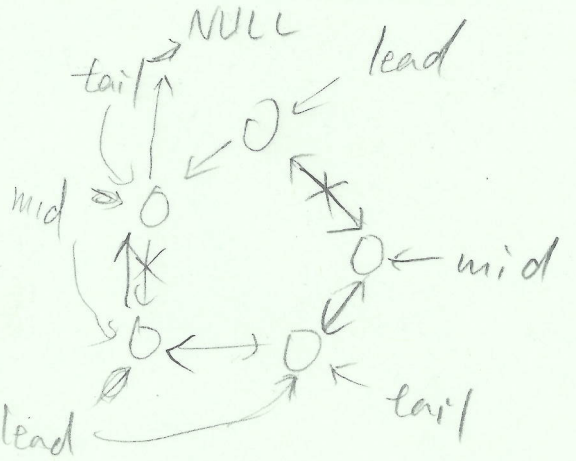
```
typedef struct cirLinklist {
        int data;
        struct cirLinklist *next;
} node;

node reverse(node* head){
    node* lead, * mid, * tail;
    mid = head;
    lead = mid -> next;
    tail = NULL;
    while (lead != NULL){
        mid -> next = tail;
        tail = mid;
        mid = lead;
        lead = lead -> next;
    }
    mid -> next = tail;
    return mid;
}
```

what if it's an empty list?

when will lead == NULL?

did not reverse the direction.

∴ tail == NULL



6.

```
treepointer copy(treepointer orig){
    if(!orig){
        NODE* temp = (NODE*)malloc(sizeof(NODE));
        temp->key = orig->key;
        temp->lchild = copy(orig->lchild);
        temp->rchild = copy(orig->rchild);
        return temp;
    }
    return NULL;
}
```

7.

```
int LargestWeightedPath(NODE* root){
    int lweight, rweight;
    if(!root) return 0;
    lweight = LargestWeightedPath(root->lchild);
    rweight = LargestWeightedPath(root->rchild);
    if(lweight > rweight)
        return root->weight + lweight;
    else
        return root->weight + rweight;
}
```

your path is not guaranteed to reach the leaf node if some weights are negative −1

8.
```
treepointer search(treepointer t, int k, int n){
    int pos = n - t->rightsize;
    if (k == pos) return t;
    if (k > pos)
      return (t->rchild, k, t->rightsize);
    else
      return (t->lchild, k, pos-1);
}
```