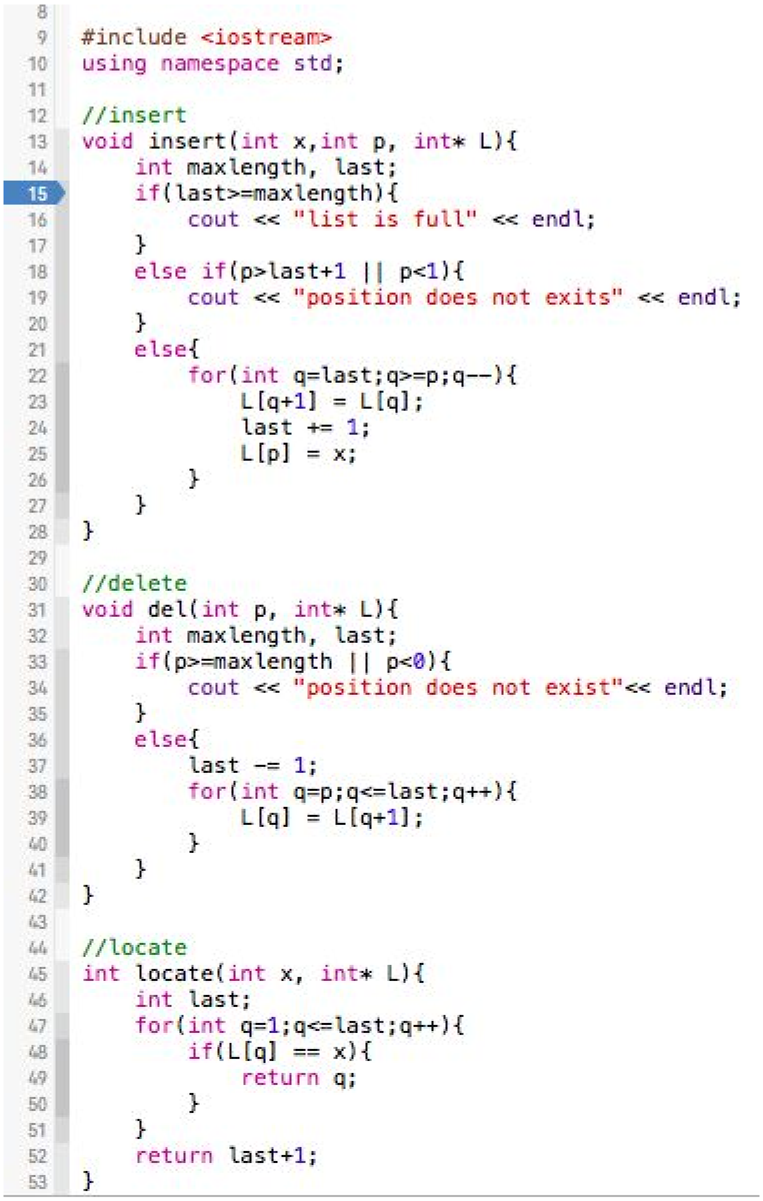
Yiyun Zhang
7/14/2016

2.2

a. array



```
#include <iostream>
using namespace std;

//insert
void insert(int x,int p, int* L){
    int maxlength, last;
    if(last>=maxlength){
        cout << "list is full" << endl;
    }
    else if(p>last+1 || p<1){
        cout << "position does not exits" << endl;
    }
    else{
        for(int q=last;q>=p;q--){
            L[q+1] = L[q];
            last += 1;
            L[p] = x;
        }
    }
}

//delete
void del(int p, int* L){
    int maxlength, last;
    if(p>=maxlength || p<0){
        cout << "position does not exist"<< endl;
    }
    else{
        last -= 1;
        for(int q=p;q<=last;q++){
            L[q] = L[q+1];
        }
    }
}

//locate
int locate(int x, int* L){
    int last;
    for(int q=1;q<=last;q++){
        if(L[q] == x){
            return q;
        }
    }
    return last+1;
}
```

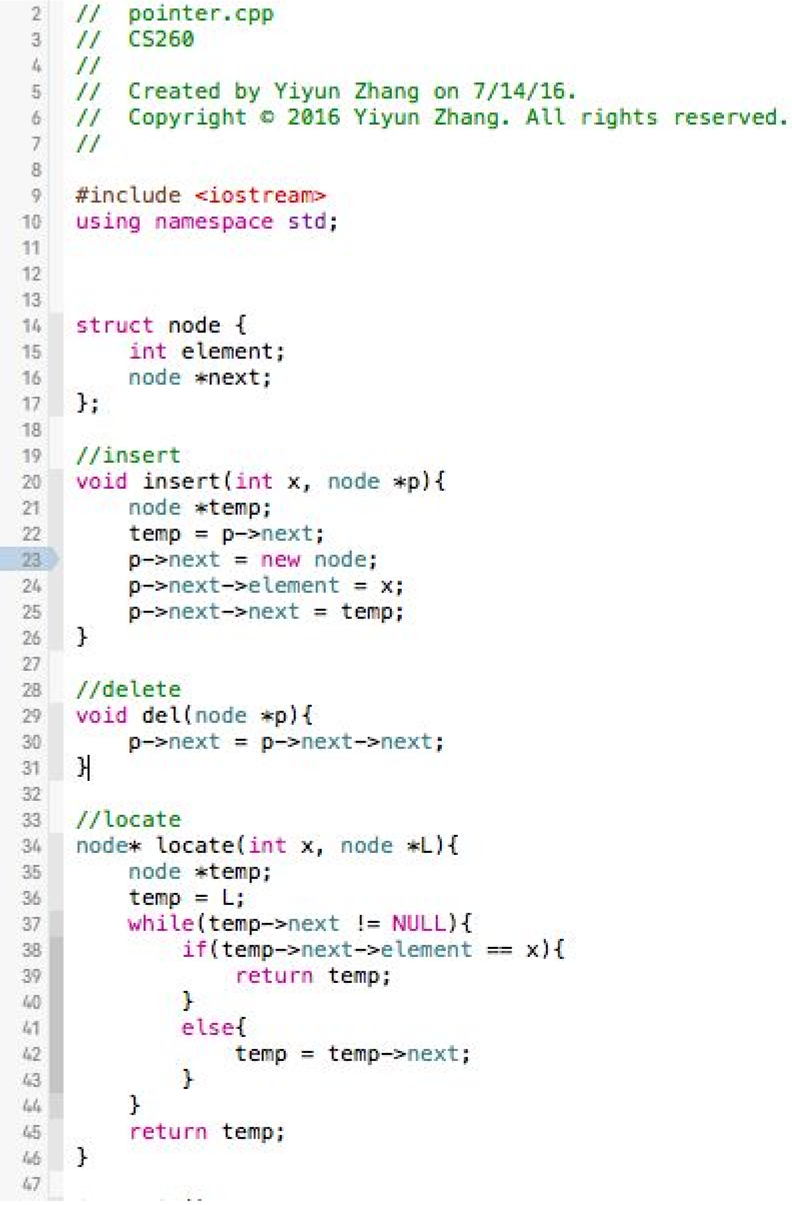
Running time:

insert:O(1)

delete:O(1)

locate:O(n)

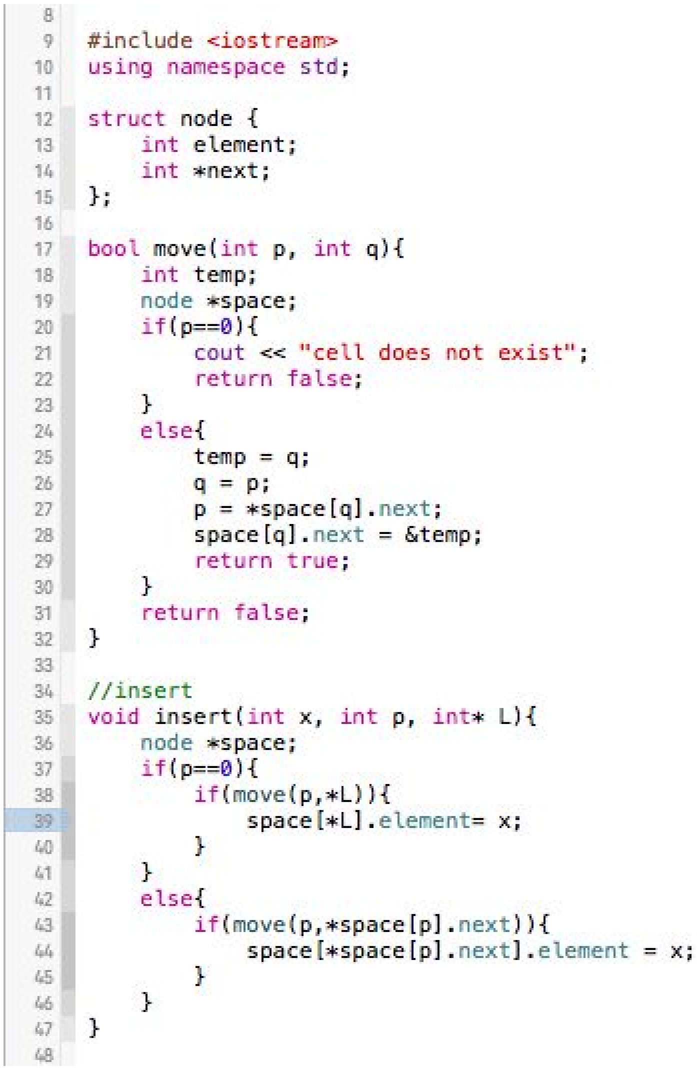
b. pointer



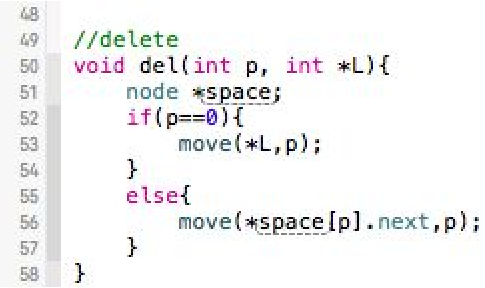
```
//  pointer.cpp
//  CS260
//
//  Created by Yiyun Zhang on 7/14/16.
//  Copyright © 2016 Yiyun Zhang. All rights reserved.
//

#include <iostream>
using namespace std;



struct node {
    int element;
    node *next;
};

//insert
void insert(int x, node *p){
    node *temp;
    temp = p->next;
    p->next = new node;
    p->next->element = x;
    p->next->next = temp;
}

//delete
void del(node *p){
    p->next = p->next->next;
}

//locate
node* locate(int x, node *L){
    node *temp;
    temp = L;
    while(temp->next != NULL){
        if(temp->next->element == x){
            return temp;
        }
        else{
            temp = temp->next;
        }
    }
    return temp;
}

```

Running time:
insert:O(1)
delete:O(1)
locate:O(n)

c. cursor-based



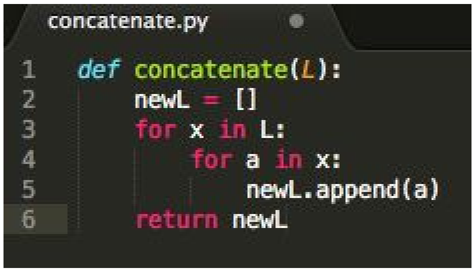
```
#include <iostream>
using namespace std;

struct node {
    int element;
    int *next;
};

bool move(int p, int q){
    int temp;
    node *space;
    if(p==0){
        cout << "cell does not exist";
        return false;
    }
    else{
        temp = q;
        q = p;
        p = *space[q].next;
        space[q].next = &temp;
        return true;
    }
    return false;
}

//insert
void insert(int x, int p, int* L){
    node *space;
    if(p==0){
        if(move(p,*L)){
            space[*L].element= x;
        }
    }
    else{
        if(move(p,*space[p].next)){
            space[*space[p].next].element = x;
        }
    }
}

```



```

//delete
void del(int p, int *L){
    node *space;
    if(p==0){
        move(*L,p);
    }
    else{
        move(*space[p].next,p);
    }
}
```

Running time:
insert:O(n)
delete:O(n)

2.4



```python
def concatenate(L):
    newL = []
    for x in L:
        for a in x:
            newL.append(a)
    return newL
```

2.11

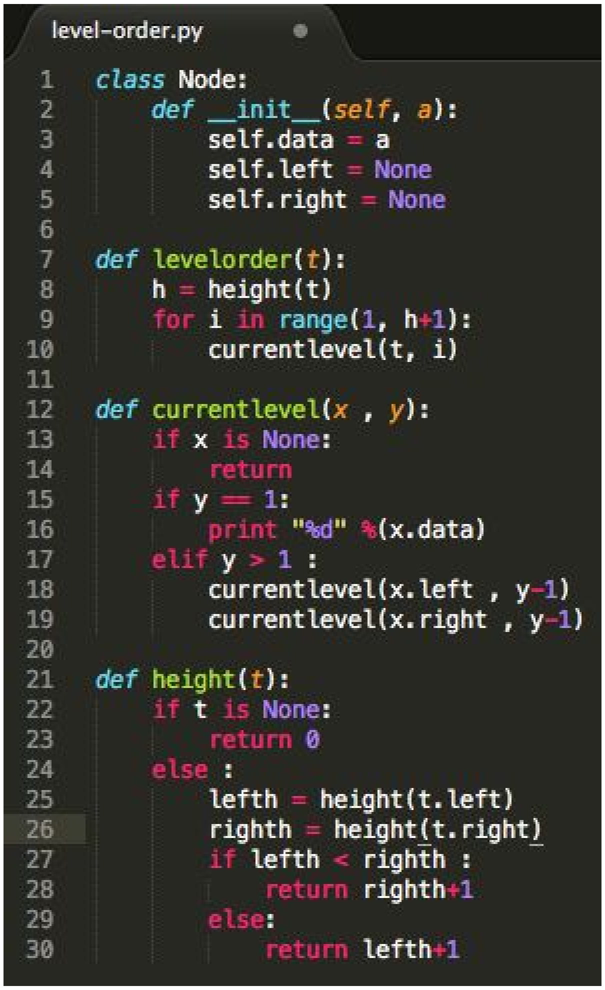
```
p := FIRST(L);                                        1
while p <> END(L) do begin                            n
        q := p;
        while q <> END(L) do begin                    n
                q := NEXT(q, L);                      n-q
                r := FIRST(L);                        1
                while r <> q do
                        r := NEXT(r, L)               n-r
        end;
        p := NEXT(p, L)                               n-p
End;
```

FIRST: 1*n*n*1 = $O(n^2)$ times

NEXT: (((n-r+n-q)*n)+n-p)*n = $O(n^3)$ times

END: n*n = $O(n^2)$ times

3.10



```python
class Node:
    def __init__(self, a):
        self.data = a
        self.left = None
        self.right = None

def levelorder(t):
    h = height(t)
    for i in range(1, h+1):
        currentlevel(t, i)

def currentlevel(x , y):
    if x is None:
        return
    if y == 1:
        print "%d" %(x.data)
    elif y > 1 :
        currentlevel(x.left , y-1)
        currentlevel(x.right , y-1)

def height(t):
    if t is None:
        return 0
    else :
        lefth = height(t.left)
        righth = height(t.right)
        if lefth < righth :
            return righth+1
        else:
            return lefth+1
```