Q1.

```
str = input('')
while str != '':
    total = 0
    a = False
    count = 0
    max = None
    min = None
    nums = str.split(',')
    ints = []
    for i in nums:
        try:
            num = float(i)
            total += num
            count += 1
            if(max is None or max < num):
                max = num
            if(min is None or min > num):
                min = num
            ints.append(num)
        except ValueError:
            print('Input is invalid.')
            a = True
        if a:
            break
    if a:
        a = False
    else:
        ints.sort()
        print('Sorted:', ints, 'Max:', max, 'Avg:', total/count, 'Min:', min)
    str = input('')
```

Q2.

```
def len():
    return stack1.len() + stack2.len()
def is_empty():
    return stack1.is_empty() and stack2.is_empty()
def top():
    if(stack2.is_empty()):
        while(stack1.is_empty() is False):
            stack2.push(stack1.pop())
    x = stack2.top()
    return x
def enqueue(x):
    stack1.push(x)
def dequeue():
```

```
    if(stack2.is_empty):
        while(stack1.is_empty() is False):
            stack2.push(stack1.pop())
    x = stack2.pop()
    return x
```

Q3.

```
def preordernext(p):
    if(p.left is not None):
        return p.left
    elif(p.right is not None):
        return p.right
    else:
        node = p
        while(node.parent is not None):
            par = node.parent
            if(par.left is node):
                if(par.right is not None):
                    return par.right
            node = par
        return None
```

O(N) worst possible time where N is number of nodes

```
def inordernext(p):
    if(p.right is not None):
        node = p.right
        while(node.left is not None):
            node = node.left
        return node
    else:
        node = p
        while(node.parent is not None):
            par = node.parent
            if(par.left is node):
                return par
            node = par
        return None
```

O(N) worst possible time where N is number of nodes

```
def postordernext(p):
    if(p.parent is None):
        return None
    elif(p.parent.right is p or p.parent.right is None):
        return p.parent
    else:
        node = p.parent.right
```

```
    while(node.left is not None):
        node = node.left
    return node
```

O(N) worst possible time where N is number of nodes

Q4.