In [4]:

```
def fib(n):
    global arr, cnt
    if n <= (len(arr)-1):
        return arr[n]
    else:
        arr.append(fib(n-1) + fib(n-2))
        return arr[n]
```

In [7]:

```
arr=[]
arr.append(0)
arr.append(1)
cnt = 1
n = 3
print(n, 'Nth of fibonacci is',fib(n))
```

```
3 Nth of fibonacci is 2
```

In [8]:

```
arr
```

Out[8]:

```
[0, 1, 1, 2]
```

In [9]:

```
def fib(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n-1) + fib(n-2)
```

In [10]:

```
fib(6)
```

Out[10]:

```
8
```

In [11]:

```
fib(4)
```

Out[11]:

```
3
```

In [29]:

```
class S_Node:
    def __init__(self,data):
        self.data = data
        self.link = None

class Link_list:
    def __init__(self):
        self.head = None
    def print_list(self):
        t = self.head
        while t:
            print(t.data)
            t = t.link
l = Link_list()
l.head = S_Node(12)
l.head.link = S_Node(34)
l.head.link.link = S_Node(26)
l.head.link.link.link = S_Node(67)
l.head.link.link.link.link = S_Node(98)


l.print_list()
```

```
12
34
26
67
98
```

In [30]:

```
def fun(head, n):
    t = head
    if (head == None):
        return
    else:
        i = 1
        while(i != n):
            t = t.link
            i += 1
        t1 = head.data
        head.data = t.data
        t.data = t1
```

In [31]:

```
fun(l.head,3)
```

In [32]:

```
l.print_list()
```

```
26
34
12
67
98
```

In [36]:

```
class S_Node:
    def __init__(self,data):
        self.data = data
        self.link = None

class Link_list:
    def __init__(self):
        self.head = None
    def print_list(self):
        t = self.head
        while t:
            print(t.data)
            t = t.link
l = Link_list()
l.head = S_Node(12)
l.head.link = S_Node(121)
l.head.link.link = S_Node(112)
l.head.link.link.link = S_Node(111)
l.head.link.link.link.link = S_Node(222)


l.print_list()
```

```
12
121
112
111
222
```

In [37]:

```
def add_l(head):
    t = head
    if head == None:
        return
    else:
        while t.link != None:
            t.data = t.data + t.link.data
            t = t.link
```

In [38]:

```
add_1(l.head)
l.print_list()
```

```
133
233
223
333
222
```

In [44]:

```
class stack:
    def __init__(self):
        self.item = []
    def display(self):
        return self.item
    def length_item(self):
        return len(self.item)
    def is_empty(self):
        return self.item == []
    def push(self,item):
        self.item.append(item)
    def pop(self):
        return self.item.pop()
    def peek(self):
        return self.item[len(self.item)-1]
    def size(self):
        return len(self.item)
```

In [45]:

```
s = stack()
s.push(2)
s.push(4)
s.push(5)
s.push(1)
s.push(12)
s.push(7)
```

In [46]:

```
s.display()
```

Out[46]:

```
[2, 4, 5, 1, 12, 7]
```

In [47]:

```
def s1(s):
    if s.is_empty():
        return
    else:
        s2 = stack()
        while not s.is_empty():
            s2.push(s.pop())
            s2.push(s.pop())
            s2.push(s2.pop()*s2.pop())
        return s2.display()
```

In [48]:

```
s1(s)
```

Out[48]:

```
[84, 5, 8]
```

In [49]:

```
class stack:
    def __init__(self):
        self.item = []
    def display(self):
        return self.item
    def length_item(self):
        return len(self.item)
    def is_empty(self):
        return self.item == []
    def push(self,item):
        self.item.append(item)
    def pop(self):
        return self.item.pop()
    def peek(self):
        return self.item[len(self.item)-1]
    def size(self):
        return len(self.item)
```

In [50]:

```
s = stack()
s.push(7)
s.push(12)
s.push(1)
s.push(5)
s.push(4)
s.push(2)
```

In [51]:

```
s.display()
```

Out[51]:

```
[7, 12, 1, 5, 4, 2]
```

In [52]:

```
def s1(s):
    if s.is_empty():
        return
    else:
        s2 = stack()
        while not s.is_empty():
            s2.push(s.pop())
            s2.push(5)
            s2.push(s2.pop()+s2.pop())
        while not s2.is_empty():
            s.push(s2.pop())
        return s.display()
```

In [53]:

```
s1(s)
```

Out[53]:

```
[12, 17, 6, 10, 9, 7]
```

In [67]:

```
class Queue:
    def __init__(self):
        self.item = []
    def display(self):
        return self.item
    def peek(self):
        return self.item[-1]
    def is_empty(self):
        return self.item == []
    def enqueue(self,item):
        self.item.insert(0,item)
    def dequeue(self):
        return self.item.pop()
    def size(self):
        return len(self.item)
```

In [72]:

```
q = Queue()
for i in range(1,6):
    q.enqueue(i*i)
print(q.display())
q.dequeue()
print(q.display())
```

```
[25, 16, 9, 4, 1]
[25, 16, 9, 4]
```

In [73]:

```
def fun(n):
    q = Queue()
    if n == 0:
        return 1
    else:
        for i in range(1,n+1):
            q.enqueue(i)
        q.display()
        t = 1
        while not q.is_empty():
            t = t * q.dequeue()
        return t
```

In [74]:

```
fun(5)
```

Out[74]:

```
120
```

In [76]:

```
def fun1(n):
    q = Queue()
    if n == 0:
        return 1
    else:
        for i in range(1,n+1):
            q.enqueue(i*i)
        q.display()
        t = 1
        while not q.is_empty():
            t = t + q.dequeue()
        return t
```

In [77]:

```
fun1(4)
```

Out[77]:

```
31
```

In [121]:

```
class S_Node:
    def __init__(self,data):
        self.data = data
        self.link = None

    def get_data(self):
        return self.data

    def set_data(self, data):
        self.data = data

    def get_next(self):
        return self.link

    def set_next(self, node):
        self.link = node
class Link_list:
    def __init__(self):
        self.head = None
    def print_list(self):
        t = self.head
        while t:
            print(t.data)
            t = t.link
l = Link_list()
l.head = S_Node(1)
l.head.link = S_Node(4)
l.head.link.link = S_Node(6)
l.head.link.link.link = S_Node(7)
l.head.link.link.link.link = S_Node(9)

l.print_list()
```

```
1
4
6
7
9
```

In [122]:

```
def fun2(head, n):
    t = head
    if head == None:
        return
    else:
        i = 1
        while (i<n):
            val = t.get_data()
            print("val",val)
            t = t.get_next()
            head.set_data(val)
            print('set_data',head.data)
            i = i+1
        t1 = head.get_data()
        t.set_data(t1)
        print('t',t.get_data())
print("_____________________________________")
```

```
_____________________________________
```

In [119]:

```
fun2(l.head, 4)
l.print_list()
```

```
val 1
set_data 1
val 4
set_data 4
val 6
set_data 6
t 6
6
4
6
6
9
```

In [123]:

```
fun2(l.head, 5)
l.print_list()
```

```
val 1
set_data 1
val 4
set_data 4
val 6
set_data 6
val 7
set_data 7
t 7
7
4
6
7
7
```

In [92]:

```
fun2(l.head, 3)
l.print_list()
```

```
4
4
4
7
7
```

In [ ]:

```

```