In [1]:
```
#1. Credit Card Validation code
import re

number=input("Enter the card number: ")

#template = r"^(4|5|6)\d{3}-?\d{4}-?\d{4}-?\d{4}"

template = re.compile(r"(^(4|5|6)\d{3})-?(\d{4})-?(\d{4})-?(\d{4})")
result = re.search(template,number)
if result:
    resultlist = result.group().split("-")
    digits = "".join(resultlist)
    for i in range(len(digits)-3):
        if digits[i]==digits[i+1] and digits[i+1]==digits[i+2] and digits[i+2]==digits[i+3]:
            valid = False
            break
        else:
            valid = True

    if valid:
        print("Card Number {} is Valid".format(number))
    else:
        print("Card Number {} is Invalid".format(number))
else:
    print("Card Number {} is Invalid".format(number))
```

```
Enter the card number: 6234-2221-1234-8900
Card Number 6234-2221-1234-8900 is Valid
```

In [2]:
```
#2. Sorting and writing Binary digits
import random
f=open("trial1.txt","r")
lines = [ i for i in f.read()]
jointed = "".join(lines).split("\n")
f.close()
# random.shuffle(jointed)
jointed.sort()
joint = "\n".join(jointed)
print(joint)
f=open("trial1.txt","w+")
f.write(joint)
f.close()
```

```
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
```

In [8]:
```
#3. Distinct link list
class Node:
    def __init__(self, num):
        self.num = num
        self._next = None

    def __repr__(self):
        return str(self.num)

class LinkedList:
    def __init__(self):
        self._head = None
        self._tail = None

    def __repr__(self):
        node = self._head
        nodes = []
        while node is not None:
            nodes.append(repr(node))
            node = node._next
#         nodes.append("None")
        return " -> ".join(nodes)

    def add_item(self,node):
        if not self._head:
            self._head = node
            self._tail = node
        else:
            self._tail._next = node
            self._tail = node

    def remove_item(self,target):
        node = self._head
#         if node.num == target:
#             self._head = node._next
#             return
        while node._next is not None:
            if node._next.num == target:
                if node._next == self._tail:
                    self._tail = node
                node._next = node._next._next
                return
    def remove_duplicate(self):
        node = self._head
        while node._next:
            if node.num == node._next.num:
                node._next = node._next._next
            else:
                node = node._next


numberLL = LinkedList()
numberLL.add_item(Node(1))
numberLL.add_item(Node(1))
numberLL.add_item(Node(1))
numberLL.add_item(Node(2))
numberLL.add_item(Node(2))
numberLL.add_item(Node(2))
numberLL.add_item(Node(3))
numberLL.add_item(Node(3))
numberLL.add_item(Node(3))
numberLL.add_item(Node(3))
numberLL.add_item(Node(5))
numberLL.add_item(Node(5))
numberLL.add_item(Node(5))
numberLL.add_item(Node(5))
numberLL.add_item(Node(5))
print("Original Linkedlist:-")
print(numberLL)
numberLL.remove_duplicate()
print("Distinct Linkedlist:-")
print(numberLL)
```

```
Original Linkedlist:-
1 -> 1 -> 1 -> 2 -> 2 -> 2 -> 3 -> 3 -> 3 -> 3 -> 5 -> 5 -> 5 -> 5 -> 5
Distinct Linkedlist:-
1 -> 2 -> 3 -> 5
```