SET – 1

Regular Programs

9. The Captain's Room

```
k = int(input())
rooms = (int(x) for x in input().split(' '))
seen = {}

for i in rooms:
    if not i in seen:
        seen[i] = 1
    else:
        seen[i] += 1

for key, val in seen.items():
    if val != k:
        print(key)
```

10. Time Delta

```
from datetime import datetime

if __name__ == '__main__':
    t = int(input())
    for _ in range(t):
        s1 = input()
        s2 = input()
        t1 = datetime.strptime(s1, "%a %d %b %Y %H:%M:%S %z")
        t2 = datetime.strptime(s2, "%a %d %b %Y %H:%M:%S %z")
        print(abs(int((t1-t2).total_seconds())))
```

11. Map and Lambda Function

```
cube = lambda x: x**3 # complete the lambda function

def fibonacci(n):
    a=[]
    x=0
    y=1
    for i in range(0, n):
        a+=[x]
        x, y = y, x+y
    return a


if __name__ == '__main__':
    n = int(input())
    print(list(map(cube, fibonacci(n))))
```

12. Validating Credit Card Numbers

```
import re

n = int(input())

pattern1 = r'^[456]\d{15}$|^[456]\d{3}-\d{4}-\d{4}-\d{4}$'

pattern2 = r'(\d)\1{3,}|(\d)\2{1}-(\d)\3{1}|-(\d)\4{3,}-'


for i in range(n):
    s = input()
    if (re.search(pattern1, s)):
        if (re.search(pattern2, s)):
            print('Invalid')
        else:
            print('Valid')
```

```
  else:
    print('Invalid')
```

13. Climbing stairs

```
class Solution:
  def climbStairs(self, n):
    if n == 1:
      return 1
    if n == 2:
      return 2
    return self.climbStairs(n - 1) + self.climbStairs(n - 2)

if __name__ == "__main__":
  n = int (input())
  result = Solution().climbStairs(n)
  print result
```

14. House Robber

```
class Solution:
 def rob(self, nums):
  if len(nums) == 1:
   return nums[0]

  dp = [0] * len(nums)
  dp[0] = nums[0]
  dp[1] = max(nums[0], nums[1])

  for i in range(2, len(nums)):
   dp[i] = max(dp[i - 1], dp[i - 2] + nums[i])

  return dp[-1]

nums=[1,2,3,1]
print(Solution().rob(nums))
```

15. Longest Palindromic subsequence

```
class Solution:
  def longestPalindromeSubseq(self, s):

    def dp(i, j):
      if i > j:
        return 0
      if i == j:
        return 1
      if s[i] == s[j]:
        return 2 + dp(i + 1, j - 1)
      return max(dp(i + 1, j), dp(i, j - 1))

    return dp(0, len(s) - 1)

s = input()
Solution().longestPalindromeSubseq(s)
```

Additional Programs

6. ginortS

```
S = input()


def s(x):
    if x.islower():
        return ord(x)
    elif x.isupper():
        return ord(x)*100000
    elif x in "13579":
        return ord(x)*10000000000
    else:
        return ord(x)*1000000000000000000

print(*sorted(S, key=s), sep='')
```

7. Text wrap

```
import textwrap

def wrap(string, max_width):
    return textwrap.TextWrapper(width=max_width).fill(text=string)


if __name__ == '__main__':
    string, max_width = input(), int(input())
    result = wrap(string, max_width)
    print(result)
```

8. Piling Up

```
import sys

def test_cubes(cubes):
    t_cube = 0

    if cubes[0] > cubes[len(cubes)-1]:
        t_cube = cubes[0]
        cubes.pop(0)
    else:
        t_cube = cubes[len(cubes)-1]
        cubes.pop(len(cubes)-1)

    while len(cubes) > 0:
        if t_cube == cubes[0]:
            t_cube = cubes.pop(0)
        elif t_cube == cubes[len(cubes)-1]:
            t_cube = cubes.pop(len(cubes)-1)
        elif (cubes[0] > cubes[len(cubes)-1]) and (t_cube >= cubes[0]):
            t_cube = cubes.pop(0)
        elif (cubes[0] < cubes[len(cubes)-1]) and (t_cube >= cubes[len(cubes)-1]):
            t_cube = cubes.pop(len(cubes)-1)
        elif (cubes[0] == cubes[len(cubes)-1]):
            t_cube = cubes.pop(0)
        else:
            return "No"
    return "Yes"

num_of_tests = input()
num_of_tests = int(num_of_tests)

for i in range(0, num_of_tests):
    input()
    cubes = list(map(int, input().split(' ')))
    print(test_cubes(cubes))
```