

## ✓ Assignment-1

### 1) Distribute Candy

```
def distribute_candies(A):
    n = len(A)

    candies = [1] * n

    for i in range(1, n):
        if A[i] > A[i - 1]:
            candies[i] = candies[i - 1] + 1

    for i in range(n - 2, -1, -1):
        if A[i] > A[i + 1]:
            candies[i] = max(candies[i], candies[i + 1] + 1)

    return sum(candies)

A = [1, 2]
result = distribute_candies(A)
print(result)
```

### 2) Best Time to Buy and Sell Stocks I

```
def max_profit(A):
    n = len(A)

    if n <= 1:
        return 0
    min_price = A[0]
    max_profit = 0

    for price in A:
        min_price = min(min_price, price)
        max_profit = max(max_profit, price - min_price)

    return max_profit

A1 = [1, 2]
A2 = [1, 4, 5, 2, 4]

result1 = max_profit(A1)
result2 = max_profit(A2)

print(result1)
print(result2)
```

1  
4

### 3)Stairs

```
def climbStairs(A):
    if A == 1:
        return 1
    if A == 2:
        return 2

    ways = [0] * (A + 1)

    ways[1] = 1
    ways[2] = 2

    for i in range(3, A + 1):
        ways[i] = ways[i - 1] + ways[i - 2]

    return ways[A]

A1 = 2
A2 = 3

result1 = climbStairs(A1)
result2 = climbStairs(A2)

print(result1)
print(result2)

2
3
```

#### 4)Kth Row of Pascal's Triangle

```
def getRow(k):
    if k < 0:
        return []

    row = [1]

    for i in range(1, k + 1):

        current_element = (row[i - 1] * (k - i + 1)) // i
        row.append(current_element)

    return row

k = 3
result = getRow(k)
print(result)

[1, 3, 3, 1]
```

#### 5) Repeat and Missing Number Array

```
def repeatedNumber(A):
    n = len(A)

    repeated, missing = 0, 0

    for i in range(n):
        index = abs(A[i]) - 1

        if A[index] > 0:
            A[index] = -A[index]
        else:
            repeated = abs(A[i])

    for i in range(n):
        if A[i] > 0:
            missing = i + 1
            break

    return [repeated, missing]

input_array = [3, 1, 2, 5, 3]
output = repeatedNumber(input_array)
print(output)

[3, 4]
```

## ✓ Assignment-2

### 6)Add One To Number

```
def add_one_to_number(digits):
    n = len(digits)
    carry = 1

    for i in range(n - 1, -1, -1):
        current_sum = digits[i] + carry
        digits[i] = current_sum % 10
        carry = current_sum // 10

    if carry:
        digits.insert(0, carry)

    return digits

input_digits = [1, 2, 3]
output_digits = add_one_to_number(input_digits)
print("Input:", input_digits)
print("Output:", output_digits)
```

```
Input: [1, 2, 4]
Output: [1, 2, 4]
```

### 7)Majority Element

```
def majority_element(nums):
    count = 0
    candidate = None

    for num in nums:
        if count == 0:
            candidate = num

        count += 1 if num == candidate else -1

    return candidate

input_array = [2, 1, 2]
result = majority_element(input_array)
print("Majority Element:", result)
```

### 8)Intersection of Linked Lists

```

class ListNode:
    def __init__(self, value=0, next=None):
        self.value = value
        self.next = next

def getIntersectionNode(headA, headB):
    def getLength(node):
        length = 0
        while node:
            length += 1
            node = node.next
        return length

    lenA, lenB = getLength(headA), getLength(headB)

    while lenA > lenB:
        headA = headA.next
        lenA -= 1

    while lenB > lenA:
        headB = headB.next
        lenB -= 1

    while headA != headB:
        headA = headA.next
        headB = headB.next

    return headA

headA = ListNode(1, ListNode(2, ListNode(3, ListNode(4, ListNode(5)))))
headB = ListNode(6, ListNode(7, headA.next.next))

intersection_node = getIntersectionNode(headA, headB)
if intersection_node:
    print("Intersection Node Value:", intersection_node.value)
else:
    print("No Intersection")

    Intersection Node Value: 3

```

## 9)Pascal Triangle

```

def generate_pascals_triangle(numRows):
    if numRows == 0:
        return []

    triangle = [[1]]

    for i in range(1, numRows):
        row = [1]
        for j in range(1, i):
            row.append(triangle[i-1][j-1] + triangle[i-1][j])
        row.append(1)
        triangle.append(row)

    return triangle

# Example usage:
numRows = 5
result = generate_pascals_triangle(numRows)
print(result)

```

```

[[1], [1, 1], [1, 2, 1], [1, 3, 3, 1], [1, 4, 6, 4, 1]]

```

## 10)Palindrome Integer

```

def is_palindrome_integer(x):

    if x < 0:
        return 0

    original_num = x
    reversed_num = 0

    while x > 0:
        digit = x % 10
        reversed_num = reversed_num * 10 + digit

```

```

        x = x // 10

    return original_num == reversed_num

input_num1 = 12121
input_num2 = 123

output1 = is_palindrome_integer(input_num1)
output2 = is_palindrome_integer(input_num2)
print(f"Input: {input_num1}, Output: {output1}")
print(f"Input: {input_num2}, Output: {output2}")

Input: 12121, Output: True
Input: 123, Output: False

```

## ✓ Assignment-3

### 11) Verify Prime

```


def is_prime(N):
    if N <= 1:
        return 0

    for i in range(2, int(N**0.5) + 1):
        if N % i == 0:
            return 0

    return 1
input_number = 7
output = is_prime(input_number)

print(f"Input: {input_number}, Output: {output}")

```

 Input: 7, Output: 1

### 12)Reverse integer

```

def reverse_integer(x):
    INT_MAX = 2**31 - 1
    INT_MIN = -2**31

    sign = 1 if x >= 0 else -1
    x = abs(x)
    reversed_num = 0

    while x > 0:
        digit = x % 10
        x = x // 10

        # Check for overflow
        if reversed_num > (INT_MAX - digit) // 10:
            return 0

        reversed_num = reversed_num * 10 + digit

    return sign * reversed_num

# Example usage:
input_num1 = 123
input_num2 = -123

output1 = reverse_integer(input_num1)
output2 = reverse_integer(input_num2)

print(f"Input: {input_num1}, Output: {output1}")
print(f"Input: {input_num2}, Output: {output2}")

Input: 123, Output: 321
Input: -123, Output: -321

```

### 13)Excel Column Title

```
def reverse_integer(x):
    INT_MAX = 2**31 - 1
    INT_MIN = -2**31

    sign = 1 if x >= 0 else -1
    x = abs(x)
    reversed_num = 0

    while x > 0:
        digit = x % 10
        x = x // 10

        if reversed_num > (INT_MAX - digit) // 10:
            return 0

        reversed_num = reversed_num * 10 + digit

    return sign * reversed_num

input_num1 = 123
input_num2 = -123

output1 = reverse_integer(input_num1)
output2 = reverse_integer(input_num2)

print(f"Input: {input_num1}, Output: {output1}")
print(f"Input: {input_num2}, Output: {output2}")
```

Input: 123, Output: 321  
Input: -123, Output: -321

#### 14)Ants on a Triangle

```
def probability_of_no_collision():
    total_outcomes = 2**3
    successful_outcomes = 2

    probability = successful_outcomes / total_outcomes
    rounded_probability = round(probability, 2)

    return rounded_probability

result = probability_of_no_collision()
print(result)
```

0.25

#### 15)Intersection Of Sorted Arrays

```
def intersect_sorted_arrays(A, B):
    result = []
    i, j = 0, 0

    while i < len(A) and j < len(B):
        if A[i] == B[j]:
            result.append(A[i])
            i += 1
            j += 1
        elif A[i] < B[j]:
            i += 1
        else:
            j += 1

    return result

input_array1 = [1, 2, 3, 3, 4, 5, 6]
input_array2 = [3, 3, 5]

output = intersect_sorted_arrays(input_array1, input_array2)
print("Output:", output)
```

Output: [3, 3, 5]

### ✓ Assignment-4

## 16) Remove Duplicates from Sorted Array

```
def removeDuplicates(A):
    if not A:
        return 0

    index = 1
    for i in range(1, len(A)):
        if A[i] != A[i - 1]:
            A[index] = A[i]
            index += 1

    return index

A1 = [1, 1, 2]
A2 = [1, 2, 2, 3, 3]

print(removeDuplicates(A1))
print(removeDuplicates(A2))
```

2  
3

## 17) Remove Duplicates from Sorted Array II

```
def removeDuplicates(nums):
    if len(nums) <= 2:
        return len(nums)

    index = 2

    for i in range(2, len(nums)):
        if nums[i] != nums[index - 2]:
            nums[index] = nums[i]
            index += 1

    return index

A1 = [1, 1, 1, 2]
A2 = [0, 0, 1, 1, 1, 1, 2, 3, 3]

print(removeDuplicates(A1))
print(A1)

print(removeDuplicates(A2))
print(A2)
```

3  
[1, 1, 2, 2]  
7  
[0, 0, 1, 1, 2, 3, 3, 3, 3]

## 18) Single Number

```
def findSingle(A):
    result = 0
    for num in A:
        result ^= num
    return result

A1 = [1, 2, 2, 3, 1]
A2 = [1, 2, 2]

print(findSingle(A1))
print(findSingle(A2))
```

3  
1

## 19) Number of 1 Bits

```
def countBits(A):  
    count = 0  
    while A:  
        count += A & 1  
        A >>= 1  
    return count
```

```
A = 11  
print(countBits(A))
```

3

## 20) Reverse Bits

```
def reverseBits(A):  
    result = 0  
    for _ in range(32):  
        result = (result << 1) | (A & 1)  
        A >>= 1  
    return result
```

```
A1 = 0  
A2 = 3
```

```
print(reverseBits(A1))  
print(reverseBits(A2))
```

0  
3221225472



