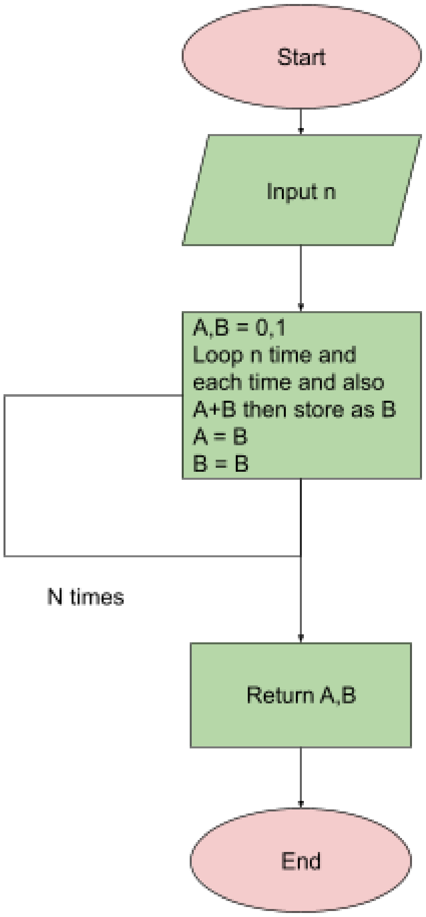
Homework 1

Simple explain english:

- Get the amount of n for the lop
- Create A,B = 0,1
- If n = 0 or 1 return A and B
- Loop through n each iteration add A and B together and store as B and keep the old B as A
- Return A and B40

Flowchart:



Pseudo Code:

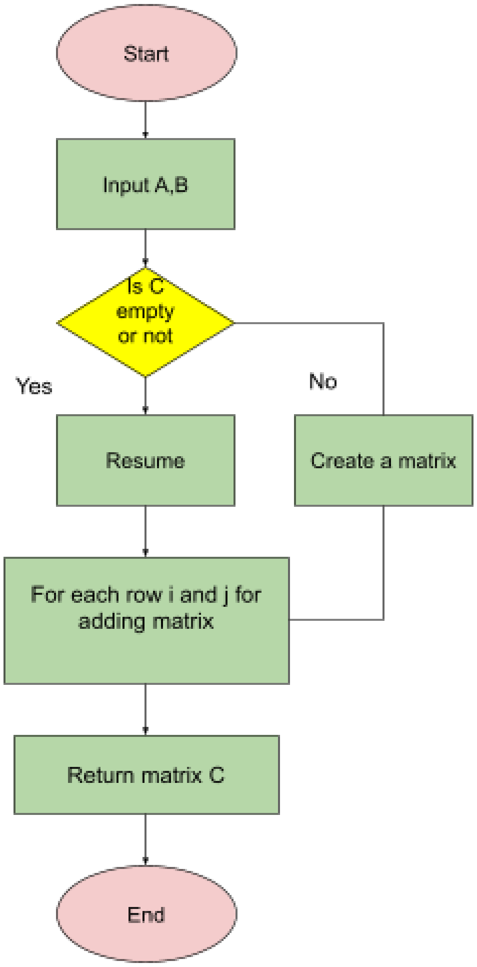
```
Input n
A,B = 0,1
If n = 0 or n =1:
        Return a,b
While 1 < n:
        a,b = b,a+b
         n - 1
Return A,B
```

Homework 2

Simple explain english:

- Create a global C
- Get input matrix from user
- Create an empty matrix and store in C
- Iterate through each number inside matrix and add them then store back in Cas A
- Return C

Flowchart:



Pseudo Code:

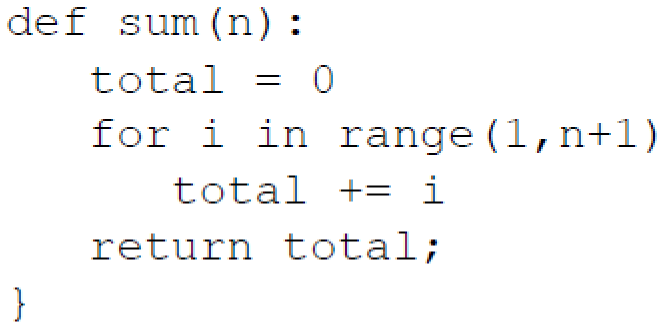
Global variable: C

```
Def Add_Matrix(A, B)
        n, m = len(A), len(A[0])
    If C is not initialized Then
        C ← new matrix of size n × m
    End If
    For i from 0 to n-1
        For j from 0 to m-1
            C[i][j] ← A[i][j] + B[i][j]
        End For
    End For
    Return C
```

Homework 3

Explain each examples in lecture 3

Ex 1

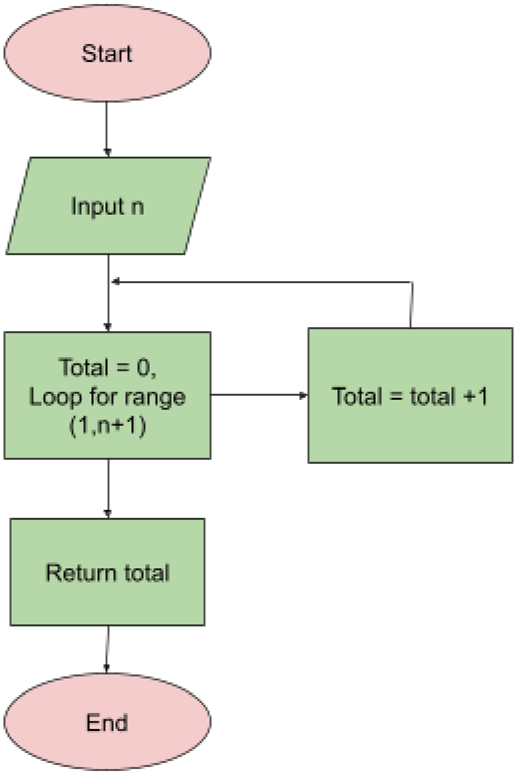


```
def sum(n):
    total = 0
    for i in range(1,n+1)
        total += i
    return total;
}
```

Simple explain english:

- Get an input n and create a variable total
- Loop through range of 1 to n+1 and each iteration add 1 to variable total
- Return total

Flowchart



Heuristic

We know that the lowest number is 1 and the largest is n+1 so we will find an average of these 2 numbers and twice it.

n = n*(1+n)/2

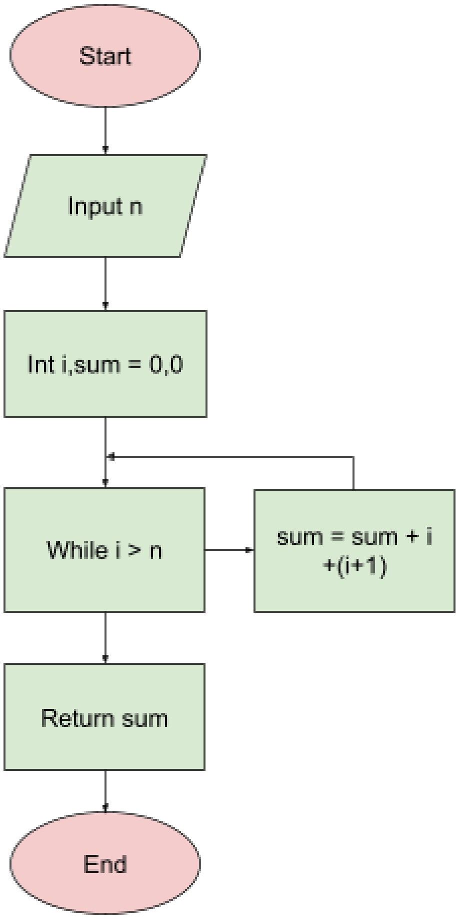
Ex 2

```
int fibonacci(n) {
   int sum = 0;
   int i = 0;
   while (i < n) {
      sum = sum + i + (i+1)
   }
   return sum;
}
```

Simple explain english:

- Get an input n and create int as sum , i = 0
- Loop for n times and in each iteration add sum with old sum plus i plus (i + 1)
- Return sum

Flowchart



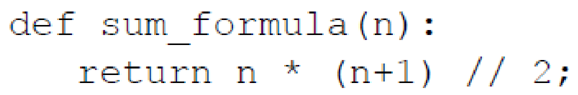
Heuristic

In this case we can see the output of n is that we add the output of n-1 and n-2 together.

Formular:

F(n)=F(n−1)+F(n−2)

Ex 3

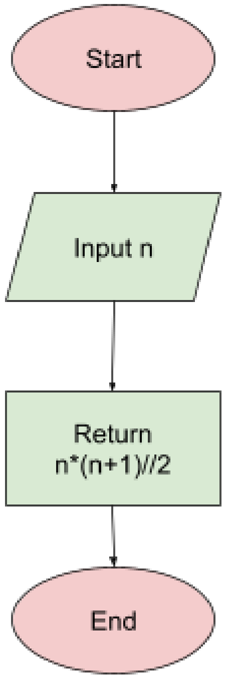


```
def sum_formula(n):
    return n * (n+1) // 2;
```

Simple explain english:

- Get an input n
- Return the the floored-division of n(n+1)

Flowchart



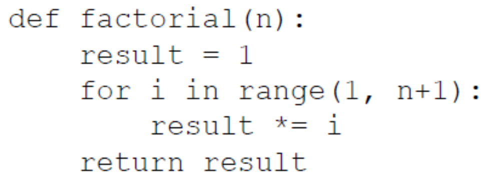
Heuristic

Instead of adding every number 1+2+3+…+n step by step,
So we will find an average of n^2

Formular

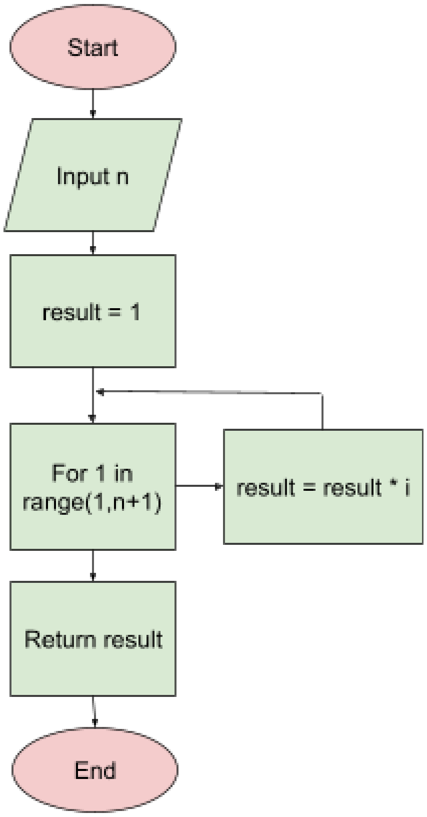
Ans = (n^2)/2

Ex 4



```
def factorial(n):
    result = 1
    for i in range(1, n+1):
        result *= i
    return result
```

Flowchart



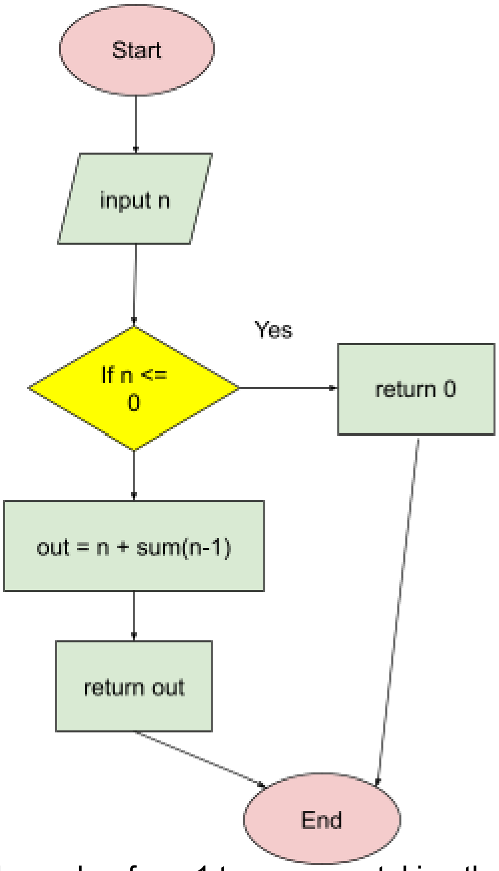
Heuristic

It just multiply 1 until n so why can't we find an average of n and twice it
Formular
Output = (n/2)^n

Ex 5

```
int sum(n) {
    if (n <= 0) {
        return 0;
    }
    out = n + sum(n-1);
    return out;
}
```

Flowchart



Heuristic

Instead of recursively adding each number from 1 to n we can taking the average value n/2 and multiplying by n,

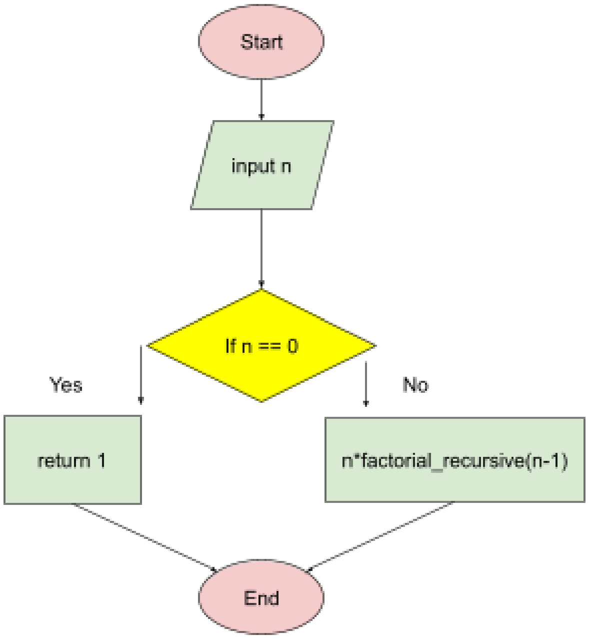
Formular

Output = (n^2)/2

Ex 6

```
def factorial_recursive(n):
    if n == 0: return 1
    return n * factorial_recursive(n-1)
```

Flowchart

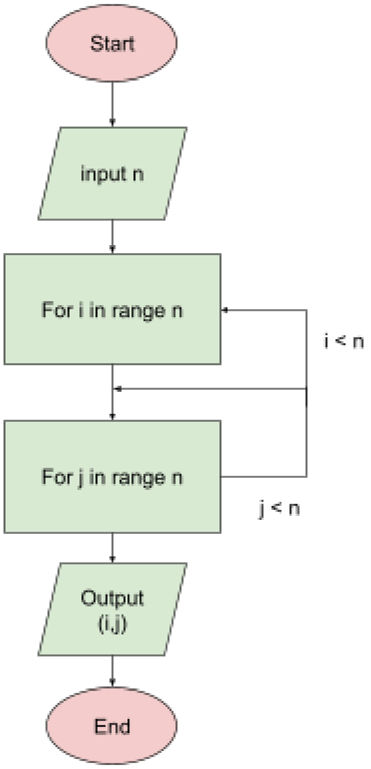


Heuristic
Instead of multiplying all numbers from 1 up to n, we can approximate factorial growth.
Factorials get large very fast

Ex 7

```
def print_pairs(n):
    for i in range(n):
        for j in range(n):
            print(i, j)
```

Flowchart



Heuristic

You can create directly by looking on n dont have to for loop

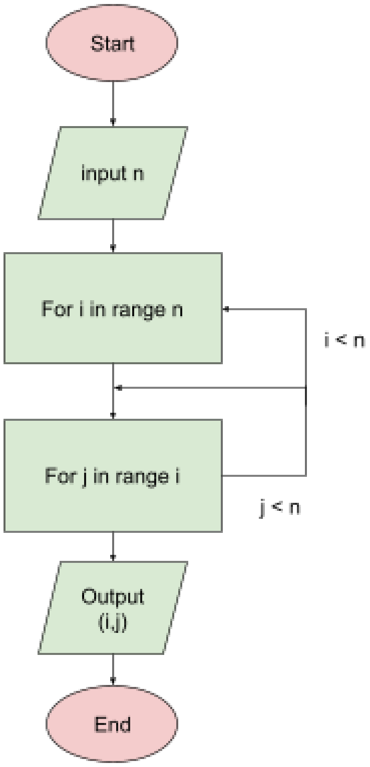
n = 2

[0,0] ,[0,1],[1,0],[1,1]

Ex 8

```
# Triangular nested loop
def triangular_loop(n):
    for i in range(n):
        for j in range(i):
            print(i, j)
```

Flowchart

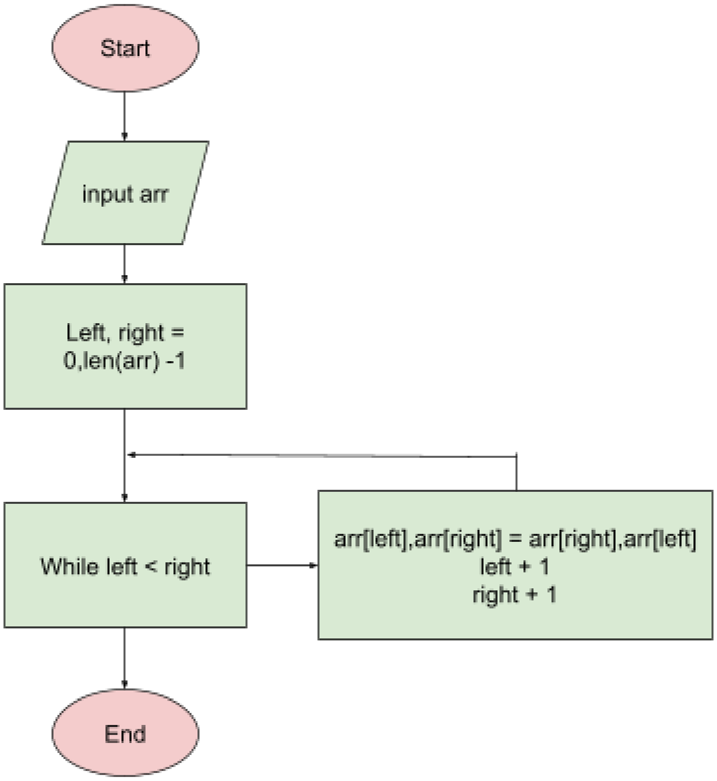


Heuristic
Same as ex7
You can create directly by looking on n dont have to for loop
n = 2
[0,0] ,[0,1],[1,0],[1,1]

Ex 9

```
# Reverse an array
def reverse(arr):
    left, right = 0, len(arr)-1
    while left < right:
        arr[left], arr[right] = arr[right], arr[left]
        left += 1
        right -= 1
```

Flowchart



Heuristic

Reversing an array means flipping the order of elements. Instead of swapping step by step, my simple guess is:

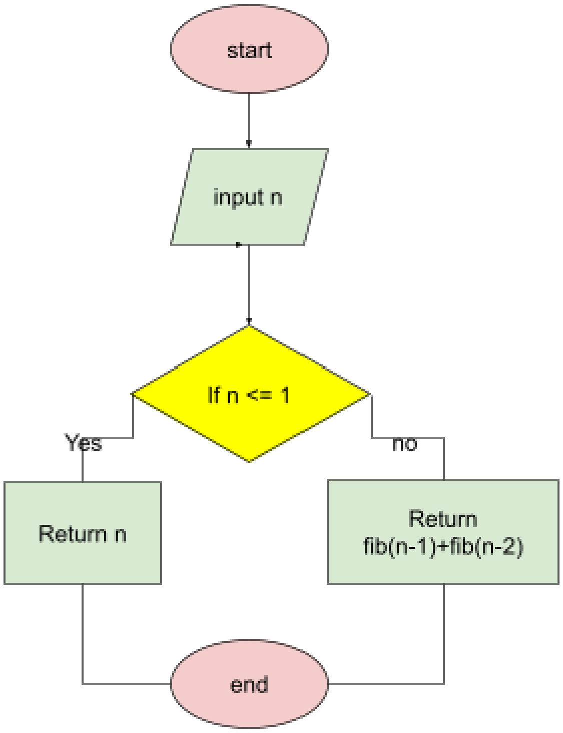
Just imagine the array backwards.

[1,2,3,4,5] to [5,4,3,2,1]

Ex 10

```
# Real Fibonacci
def fib(n):
    if n <= 1:
        return n
    return fib(n-1) + fib(n-2)
```

Flowchart



Heuristic

The Fibonacci sequence is made by adding the last two numbers to get the next one.

- Start with 0 and 1
- Then each new number ≈ sum of the two before it.

Example: 0, 1, 1, 2, 3, 5, 8, 13 …