- Username : class-1
- Password :

- Check VS 2019 whether can use
- We will start our course in 18:30
- we will start demonstrate the exercises at 19:15.
- Do not use scanf_s
- Please make sure the TA has recorded your exercise score here before leaving.

Exercise Submission Format

Format:

- xxxxxxxxx_ex_w07.zip
  - xxxxxxxxx_ex_01.cpp
  - xxxxxxxxx_ex_02.cpp

xxxxxxxxx is your student ID

Homework Submission Format

Format:

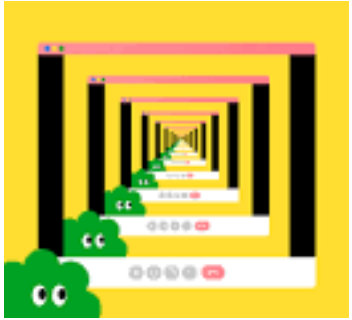
- xxxxxxxxx_hw_w07.zip
  - xxxxxxxxx_hw_01.cpp
  - xxxxxxxxx_hw_02.cpp

xxxxxxxxx is your student ID

Recursive Function

- Introduction to Computers and Programming
- Week7 TA Course 2023/10/24

What is recursion?



Calculate Factorials

- $f(n) = n! = n \cdot (n-1) \cdot (n-2) \ldots 1$

Input:n=5
Output:ans =120

5*4*3*2*1

iterative way(for loop)

```
void main() {
    int n = 5;
    int ans = 1;
    for (int i = n; i >= 1;i--) {
        ans = ans * i;

    }
    printf("%d", ans);
}
```

Calculate Factorials

- f(n) = n! = n · (n-1) · (n-2) … 1

Input:5
Output:120

f(5)=5*~~f(4)~~
~~f(4)~~=4*~~f(3)~~
~~f(3)~~=3*~~f(2)~~
~~f(2)~~=2*~~f(1)~~
~~f(1)~~=1

Output = f(5)=5*4*3*2*1

recursive way(function)

```
int factorials(int n) {
    if (n == 1) {
        return 1;
    }
    else {
        return n * factorials(n - 1);

    }
}

void main() {
    int n = 5;
    int ans = 1;
    /* ... */
    printf("%d", factorials(n));
}
```

Direct recursion/Indirect Recursion

- Direct recursion
  - Direct recursion is the type of recursion in which a function directly calls itself within its own block of code.

```
int factorials(int n) {
    if (n == 1) {
        return 1;
    }
    else {
        return n * factorials(n - 1);

    }
}
```

Direct recursion/Indirect Recursion

- Indirect Recursion
  - In indirect recursion, a function calls another function which then calls the first function again.

```
int A() {
// CODE
    //...
    B();
    //...
}
```

```
int B() {
// CODE
    //...
    A();
    //...
}
```

Base Case / General Case

- **Base Case**
  - Return without making any recursive call. (halting case)
  - You must have at least one base case in a recursive call.

```
int factorials(int n) {
    if (n == 1) {
        return 1;
    }
    else {
        return n * factorials(n - 1);
    }
}
```

Base Case / General Case

- **General Case**
  - Where the recursive call takes place.
  - Happens most of the time.

```
int factorials(int n) {
    if (n == 1) {
        return 1;
    }
    else {
        return n * factorials(n - 1);

    }
}
```

Example1

Input:n=1

Output:45

f(1)=1+~~f(2)~~

~~f(2)~~=2+~~f(3)~~

~~f(3)~~=3+~~f(4)~~

.

.

~~f(8)~~=8+~~f(9)~~

~~f(9)~~=9+~~f(10)~~

~~f(10)~~=0

Output =

f(1)=1+2+3+…+8+9+0=45

```c
int ex1(int n) {
    if (n == 10) {
        return 0;
    }
    else {
        return n + ex1(n +1);

    }
}

void main() {
    int n = 1;
    /* ... */
    printf("%d", ex1(n));
}
```

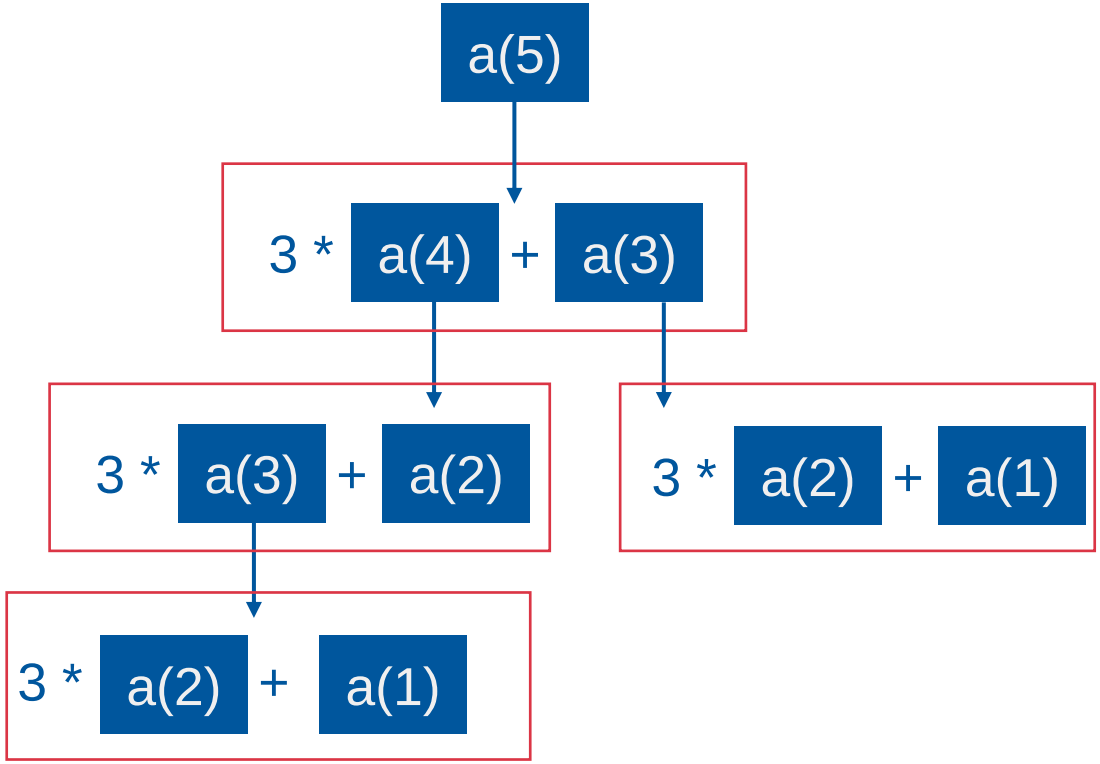
Example2

a5 = 3*(3*(3*2 +3) +2) +(3*2)+3=96

$a_n = 3a_{n-1} - a_{n-2}$; $a_1 = 3$, $a_2 = 2$

$a_5 = ?$ $a_5 = 96$

```
int a(int n) {
    if (n == 1) {
        return 3;
    }
    if (n == 2) {
        return 2;
    }
    return 3*a(n-1) + a(n -2);
```



Example3

Given a number 3, print out all the array with length 3 which every element consist of 1~3.

```
111
112
113
121
122
123
131
132
133
211
...
333
```

```c
for (int i = 1; i <= 3;i++) {
    for (int j = 1; j <= 3;j++) {
        for (int k = 1; k <= 3;k++) {
            printf("%d%d%d\n", i, j, k);
        }
    }
}
```

Example3

Given a number n, print out all the array with length n which every element consist of 1~n.

Hint

How does human produce sequence in this question?

(general case)

(base case)

- Start from 1 to n, put the number into array.
- If array length is equal to n, print array elements.

You can find many different ways of writing the recursion problem on the Internet.

Example3

Given a number 3, print out all the array with length 3 which every element consist of 1~3.

```
111
112
113
121
122
123
131
132
133
211
...
333
```

```c
void generatePermutations(int n, int k, int sequence[], int current) {
    if (current == k) {
        for (int i = 0; i < k; i++) {
            printf("%d", sequence[i]);
        }
        printf("\n");
        return;
    }

    for (int i = 1; i <= n; i++) {
        sequence[current] = i;
        generatePermutations(n, k, sequence, current + 1);
    }
}
int main() {
    int n = 3;
    int k = 3;
    int sequence[3];
    generatePermutations(n, k, sequence, 0);
```

Exercise

Please use the recursive function to complete the exercise

Exercise1

0, 1, 1, 2, 3, 5, 8, 13...

Given an integer n, output the nth Fibonacci number in recursive way.

Input

n (0 ≤ n ≤ 30)

Output

The nth Fibonacci number.

Example Input 1

5

Example Input 2

18

Example Output 1

5

Example Output 2

2584

Exercise2

- There is an array a~z(array={‘a’,’b’,’c’…….’z’}), and then input n, which means I want to print out n letters from the beginning.

Input

n (1 ≤ n ≤ 26)

Output
n letters

Example Input 1

3

Example Output 1

abc

Example Input 1

5

Example Output 1

abcde