CS 111
- Kangmei Yang
A function that calls itself to do the same task to solve a smaller/simpler version of the current task.

Same idea as:
```cpp
int max(int a, int b){
    if (a > b) return a;
    return b;
}
```
```cpp
int max(int a, int b, int c){
    return max(max(a, b),c);
}
```
```cpp
int main(){
    cout << max(3,5,4);
    return 0;
}
```
Recursion

Base Case
Terminate case (Simplest form of the input)
Done the task in one simple step

Iterative Step
Do the task at current/last step
A recursive call, the argument of the recursive call
go towards base case
Factorial

A function called factorial which takes an integer and compute the factorial of that number.

\[ 4! = 4 * 3 * 2 * 1; \]
\[ 4! = 4 * 3! \]
\[ 3! = 3 * 2! \]
\[ 2! = 2 * 1! \]
\[ 1! = 1 \]

```cpp
int factorial(int x)
{
    if (x == 1) return 1;
    return x * factorial(x - 1);
}

... int ans = factorial(4);  // 24
```
Function 1

//Write a recursive function printWithHyphen
//which takes a positive integer x, print each digit
//in x separates by a –
//Sample: 1234 output 1-2-3-4
//Sample: 4 output 4
//Sample: 1 output 1
//Sample: 12 output 1-2
//Sample: 123 output 1-2-3

void printWithHyphen(int x)
{
    //base/terminate case
    if (x < 10) {
        cout << x;
        return;
    }
    printWithHyphen(x / 10);
    cout << "-" << x % 10;
}
void printWithHyphen(int x){
    //base/terminate case
    if (x < 10) {
        cout << x;
        return;
    }
    printWithHyphen(x / 10);
    cout << "-" << x % 10;
}

int main(){
    printWithHyphen(1234);
    return 0;
}

1 - 2 - 3 - 4
Function 2

//Write a recursive function removeFirstDigit
//which takes a positive integer x
//return back x with all digits except first digit
//Sample: 1234 return 234
//Sample: 4 return 0
//Sample: 1 return 0
//Sample: 12 return 2
//Sample: 123 return 23

int removeFirstDigit(int x) {
    //base / x in simplest form
    if (x < 10) {
        return 0;
    }
    int ans = removeFirstDigit(x / 10);
    //if this function works correctly
    //what should above recursive call do?
    return ans * 10 + x % 10;
}
Function 3

//Write a recursive function countDigit
//which takes a positive integer x
//return back how many digits in x
//Sample:1524 return 4
//Sample:4 return 1
//Sample:1 return 1
//Sample:15 return 2
//Sample:152 return 3

int removeFirstDigit(int x){
//base / x in simplest form
    if (x < 10)
        return 1;
    int ans = removeFirstDigit (x / 10 );
//if this function works correctly
//what should above recursive call do?
    return ans + 1;
}
Function 4

//Write a recursive function gcd
//which takes two positive integer x, y
//we assume that x is greater or equal to y
//return back greatest common factor in x and y
//Sample:121, 11 return 11
//Sample:91, 65
//Euclid said: the gcd doesn’t change if the larger
//number replaced by the difference with smaller
//number, thus (91 – 65) has the same gcd
//Procedure: replace larger number with
//the remainder until remainder is 0

```c
int gcd(int x, int y){
    if (x % y == 0)
        return y;
    return gcd(y, x % y );
}
```
int gcd(int x, int y){
    if (x % y == 0)
        return y;
    return gcd(y, x % y);
}

int main(){
    cout << gcd(91, 65) << 13;
    return 0;
}