```c
/* Return length of number */
long long numLen(long long num)
{
    if (num >= 0 && num < 10) return 1;
    if (num >= 10 && num < 100) return 2;
    if (num >= 100 && num < 1000) return 3;
    if (num >= 1000 && num < 10000) return 4;
    if (num >= 10000 && num < 100000) return 5;
    if (num >= 100000 && num < 1000000) return 6;
    if (num >= 1000000 && num < 10000000) return 7;
    if (num >= 10000000 && num < 100000000) return 8;
    if (num >= 100000000 && num < 1000000000) return 9;
    if (num >= 1000000000 && num < 10000000000) return 10;
    if (num >= 10000000000 && num < 100000000000) return 11;
    if (num >= 100000000000 && num < 1000000000000) return 12;
    if (num >= 1000000000000 && num < 10000000000000) return 13;
    if (num >= 10000000000000 && num < 100000000000000) return 14;
    if (num >= 100000000000000 && num < 1000000000000000) return 15;
    if (num >= 1000000000000000 && num < 10000000000000000) return 16;
    if (num >= 10000000000000000 && num < 100000000000000000) return 17;
    if (num >= 100000000000000000 && num < 1000000000000000000) return 18;
}
```

CS101: Introduction to Programming

Venkatesh Vinayakarao
Department of Computer Science and Engineering
IIIT Sri City, Chittoor.
venkatesh.v@iiits.in

Revision - What is the Output?

```
1  int main()
2  {
3      int num,num2;
4      scanf("%d%i",&num,&num2);
5
6      printf("%d\t%d",num,num2);
7      return 0;
8  }
```

Agenda

- Recursion
  - The Concept
  - Analyzing Recursive Programs

Recursion

Number Series

- Predict the next number

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

Fibonacci Series

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
0	1	1	2	3	5	8	13	21	34	55

$$F_n = F_{n-1} + F_{n-2}$$

Recursion

- A function can call itself!

```
void recursion() {
    recursion();
}

int main() {
    recursion();
}
```

This recursive program causes infinite calls and hence crashes.

Fibonacci Series

```c
#include<stdio.h>
int fib(int n)
{
    if (n <= 1)
        return n;
    return fib(n-1) + fib(n-2);
}

int main ()
{
   int n = 10;
   printf("%d", fib(n));

   return 0;
}
```

Factorial

A natural definition of factorial.

$$x! \begin{cases} 1 \text{ when } x == 1 \\ x * (x-1)! \text{ when } x > 1 \end{cases}$$

Factorial

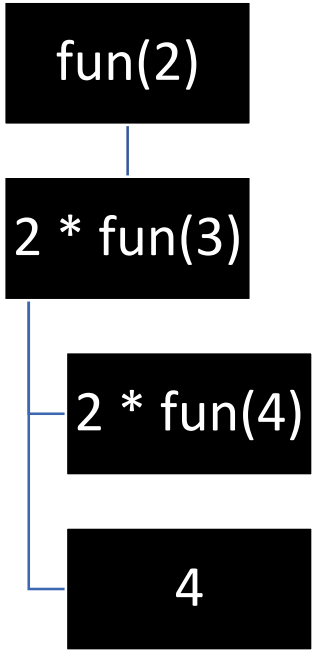
- 5! = 5 * 4 * 3 * 2 * 1 = 120
- 5! = 5 * 4!
  - = 5 * 4 * 3!
  - = 5 * 4 * 3 * 2!
  - = 5 * 4 * 3 * 2 * 1!
  - = 5 * 4 * 3 * 2 * 1
  - = 120

Factorial

```
int factorial(int x) {

   int f;

   if(x == 1)
      return 1;
   else
      f = x * factorial(x - 1);

   return f;
}

int  main() {
   int i = 5;
   printf("Factorial of %d is %d\n", i, factorial(i));
   return 0;
}
```

What is the Output?

```c
#include <stdio.h>

int fun(int n)
{
    if (n == 4)
        return n;
    else return 2*fun(n+1);
}


int main()
{
   printf("%d ", fun(2));
   return 0;
}
```



2 * 2 * 4 = 16

What is the Output?

```c
#include <stdio.h>

int fun(int x, int y)
{
  if (x == 0)
    return y;
  return fun(x - 1,  x + y);
}

int main()
{
   printf("%d ", fun(4,3));
   return 0;
}
```

fun(4,3)
fun(3,7)
fun(2,10)
fun(1, 12)
fun(0, 13)
13

Another Example

```c
#include <stdio.h>

void fun(int n)
{
  if (n == 0)
    return;

  printf("%d", n%2);
  fun(n/2);
}

int main()
{
   fun(25);
   return 0;
}
```

fun(25)
1 fun(12)
0 fun(6)
0 fun(3)
1 fun(1)
1 fun(0)

10011 is printed.

What Does This Function Do?

```
int fun(int x, int y)
{
    if (y == 0)    return 0;
    return (x + fun(x, y-1));
}
```

Assume y > 0.
fun(x, y)
x + fun(x, y-1)
x = x, y = y -1
Assume y is still > 0
x + fun(x, y-2)
Assume y = 0 now.
0

So, answer = 0
= x + 0
= x + x + 0

For any y, we get x * y.

What is the Output?

```c
#include<stdio.h>
void print(int n)
{
    if (n > 4000)
        return;
    printf("%d ", n);
    print(2*n);
    printf("%d ", n);
}

int main()
{
    print(1000);
    getchar();
    return 0;
}
```

print(1000)
1000 print(2000) 1000 : 1000 2000 4000 4000 2000 1000
2000 print(4000) 2000 : 2000 4000 4000 2000
4000 print(8000) 4000 : 4000 4000

Answer
1000 2000 4000 4000 2000 1000

What is the Output?

```c
#include <stdio.h>
void fun(int n)
{
    if(n > 0)
    {
        fun(n-1);
        printf("%d ", n);
        fun(n-1);
    }
}

int main()
{
    fun(4);
    return 0;
}
```

```
1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
Process returned 0 (0x0)   execution
Press any key to continue.
```

What is the Output?

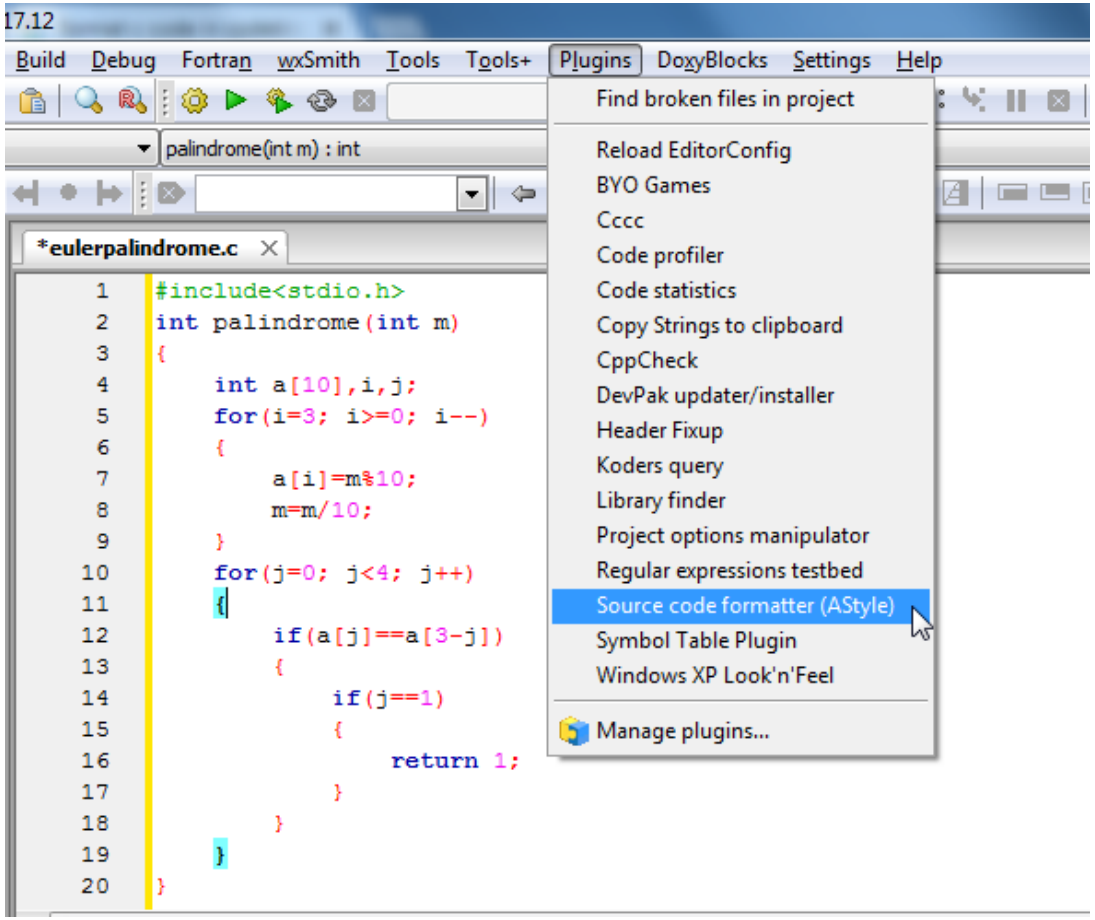
```c
#include<stdio.h>
int fun(int count)
{
    printf("%d\n", count);
    if(count < 3)
    {
      fun(fun(fun(++count)));
    }
    return count;
}

int main()
{
    fun(1);
    return 0;
}
```

```
1
2
3
3
3
3
3

Process returned 0 (0x0)
Press any key to continu
```

Formatting Your Code

eulerpalindrome.c

```c
#include<stdio.h>
int palindrome(int m)
{
int a[10],i,j;
for(i=3;i>=0;i--)
{
a[i]=m%10;
m=m/10;
}
for(j=0;j<4;j++)
{
if(a[j]==a[3-j])
{
if(j==1)
{
return 1;
}
}
}
}
```



So, what did we discuss?

- Recursion

Questions?