

# Call Stack

# Call Stack

- When you call a function, the system sets aside space in memory for that function to do its necessary work.
  - We frequently call such chunks of memory **stack frames** or **function frames**.
- More than one function's stack frame may exist in memory at a given time. If `main()` calls `move()`, which then calls `direction()`, all three functions have open frames.

# Call Stack

- These frames are arranged in a **stack**. The frame for the most-recently called function is always on the top of the stack.
- When a new function is called, a new frame is **pushed** onto the top of the stack and becomes the active frame.
- When a function finishes its work, its frame is **popped** off of the stack, and the frame immediately below it becomes the new, active, function on the top of the stack. This function picks up immediately where it left off.

# Call Stack

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

```
int main(void)
{
    printf("%i\n", fact(5));
}
```

main()

# Call Stack

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

```
int main(void)
{
    printf(“%i\n”, fact(5));
}
```

printf()

main()

# Call Stack

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

```
int main(void)
{
    printf(“%i\n”, fact(5));
}
```



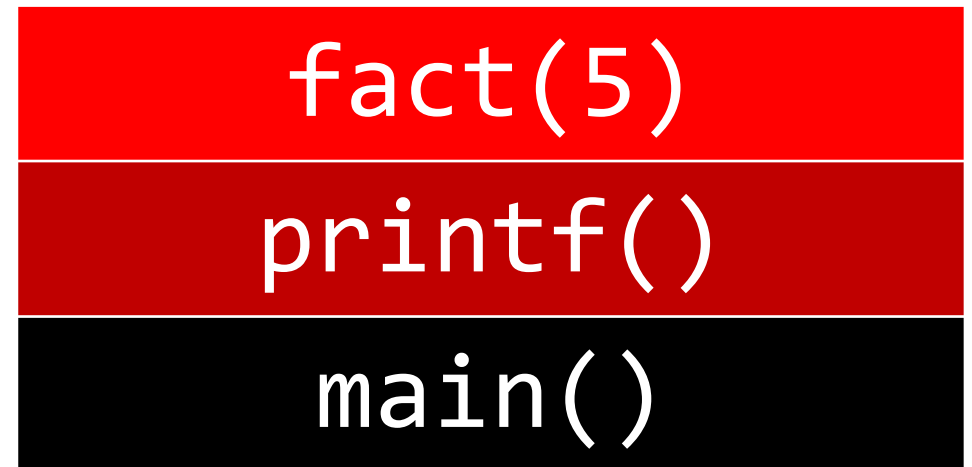
printf()

main()

# Call Stack

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

```
int main(void)
{
    printf(“%i\n”, fact(5));
}
```



# Call Stack

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

```
int main(void)
{
→   printf(“%i\n”, fact(5));
}
```

fact(5)

printf()

main()

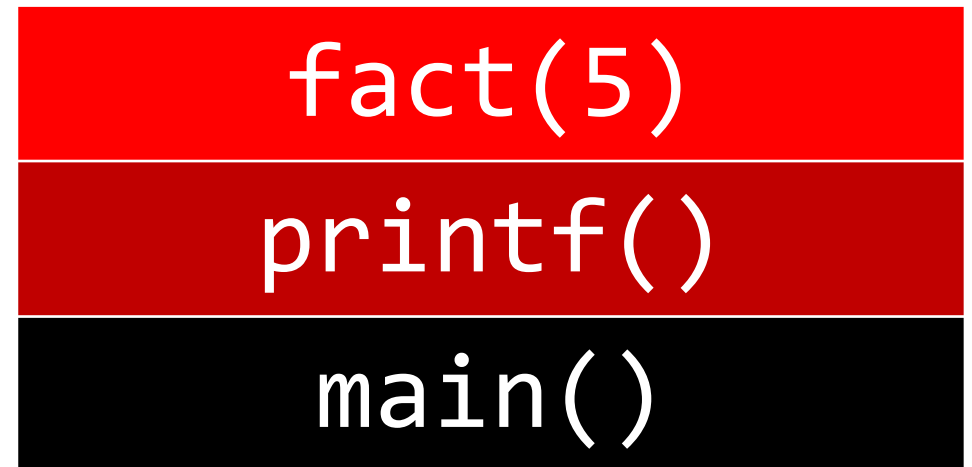


# Call Stack

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



```
int main(void)
{
    printf(“%i\n”, fact(5));
}
```



# Call Stack

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

```
int main(void)
{
    printf(“%i\n”, fact(5));
}
```

fact(4)

fact(5)

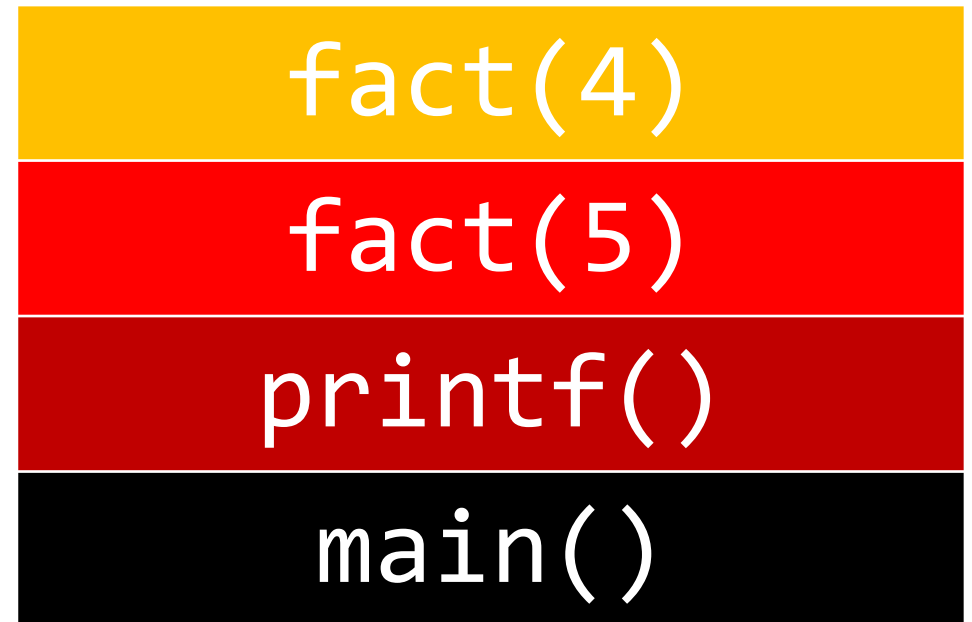
printf()

main()

# Call Stack

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

```
int main(void)
{
    printf("%i\n", fact(5));
}
```

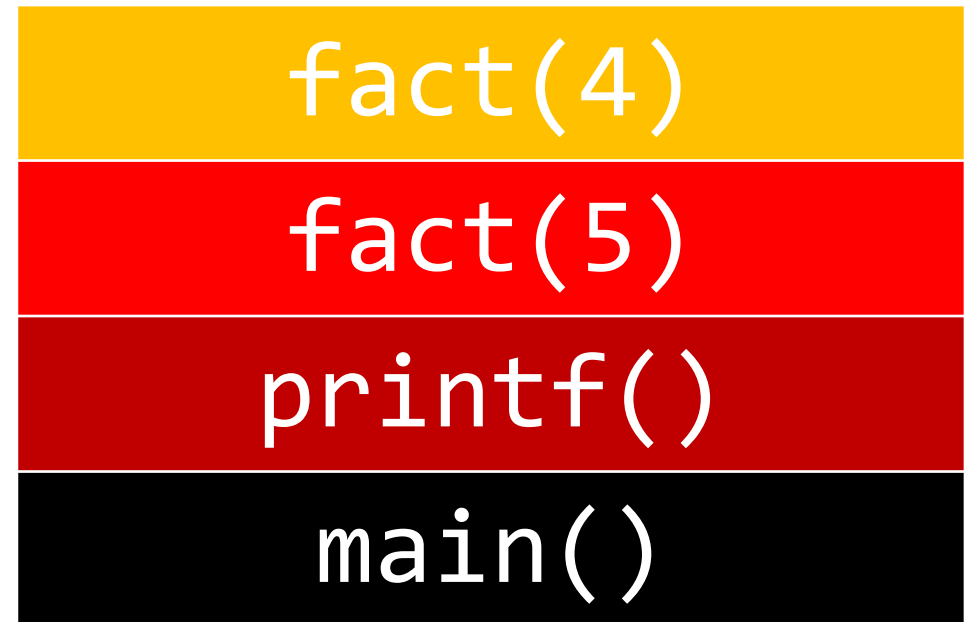


# Call Stack

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



```
int main(void)
{
    printf(“%i\n”, fact(5));
}
```

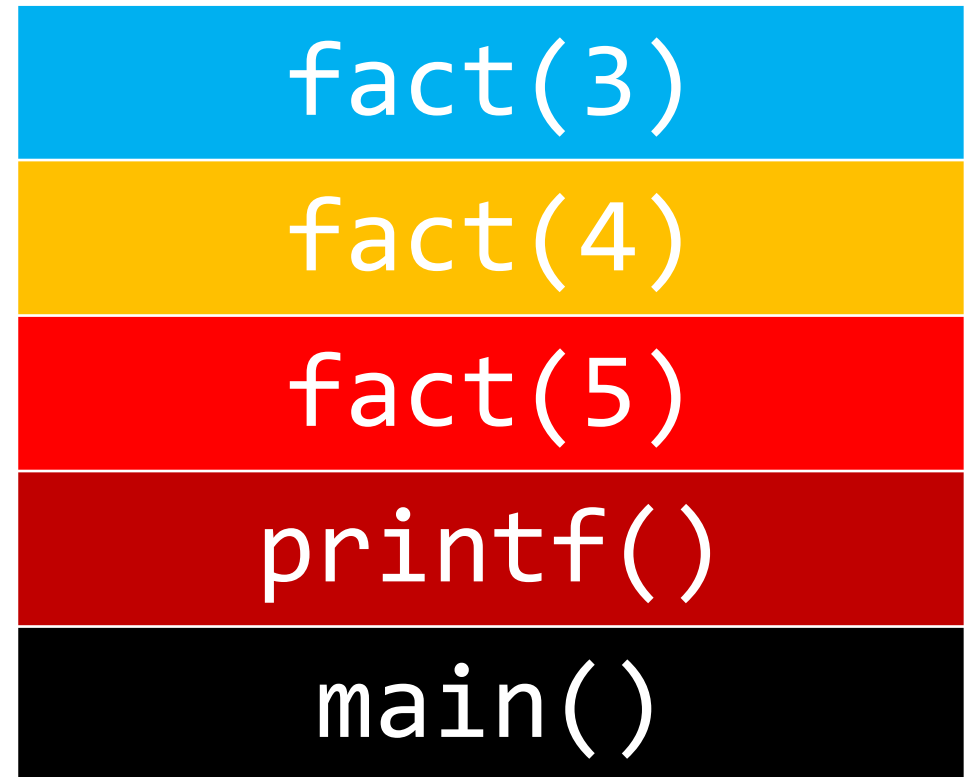


# Call Stack

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



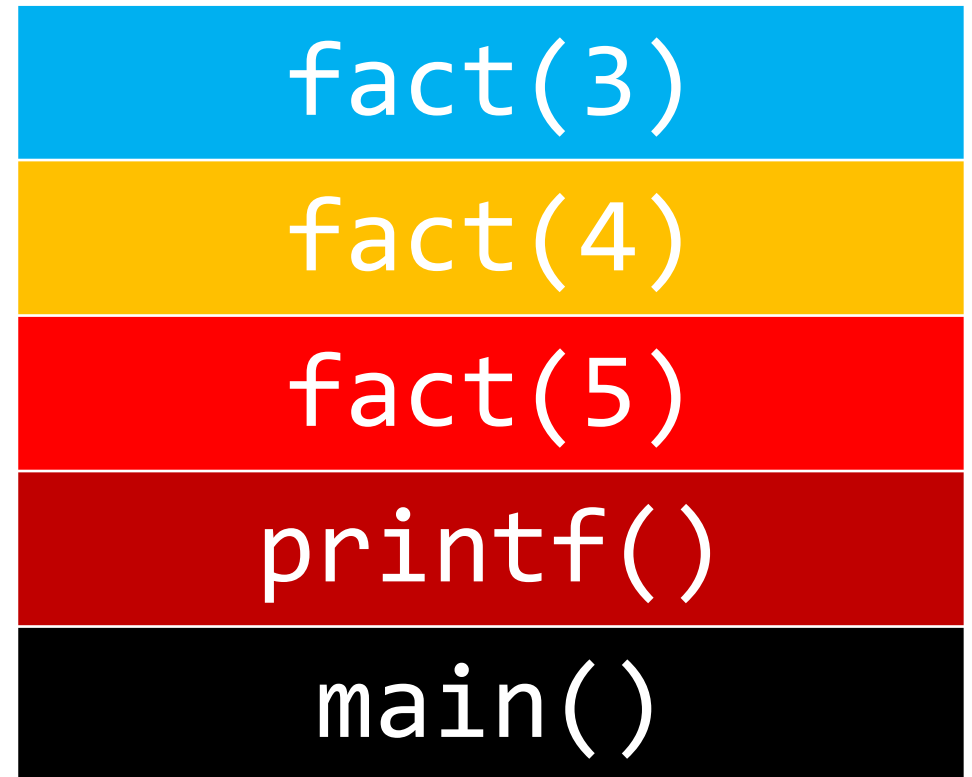
```
int main(void)
{
    printf(“%i\n”, fact(5));
}
```



# Call Stack

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

```
int main(void)
{
    printf("%i\n", fact(5));
}
```

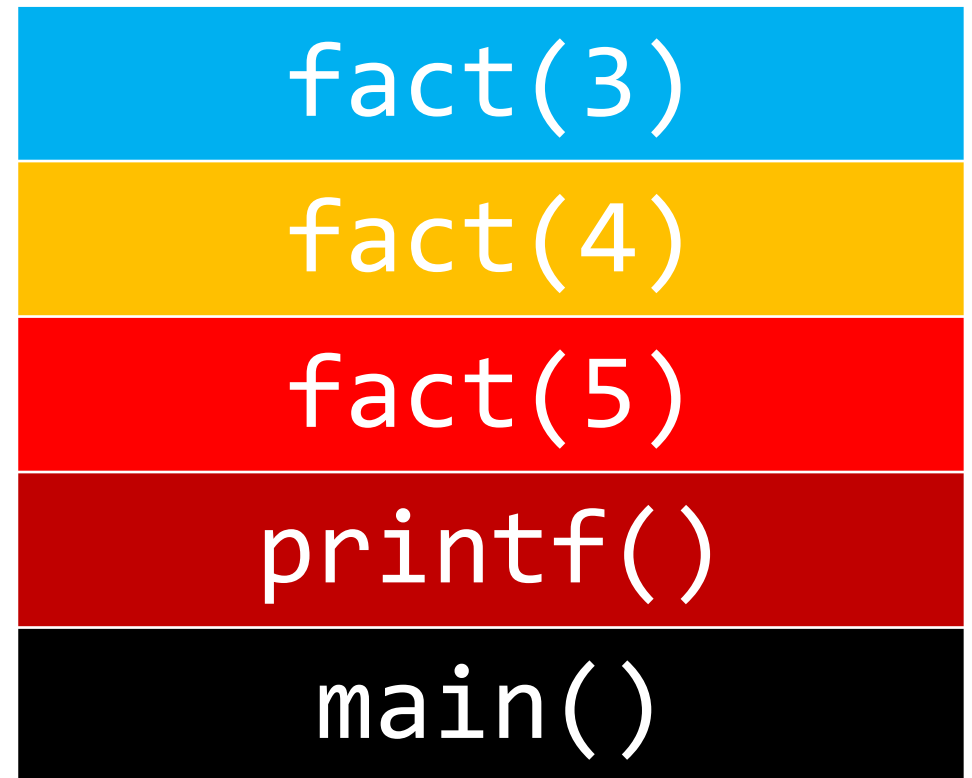


# Call Stack

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



```
int main(void)
{
    printf(“%i\n”, fact(5));
}
```

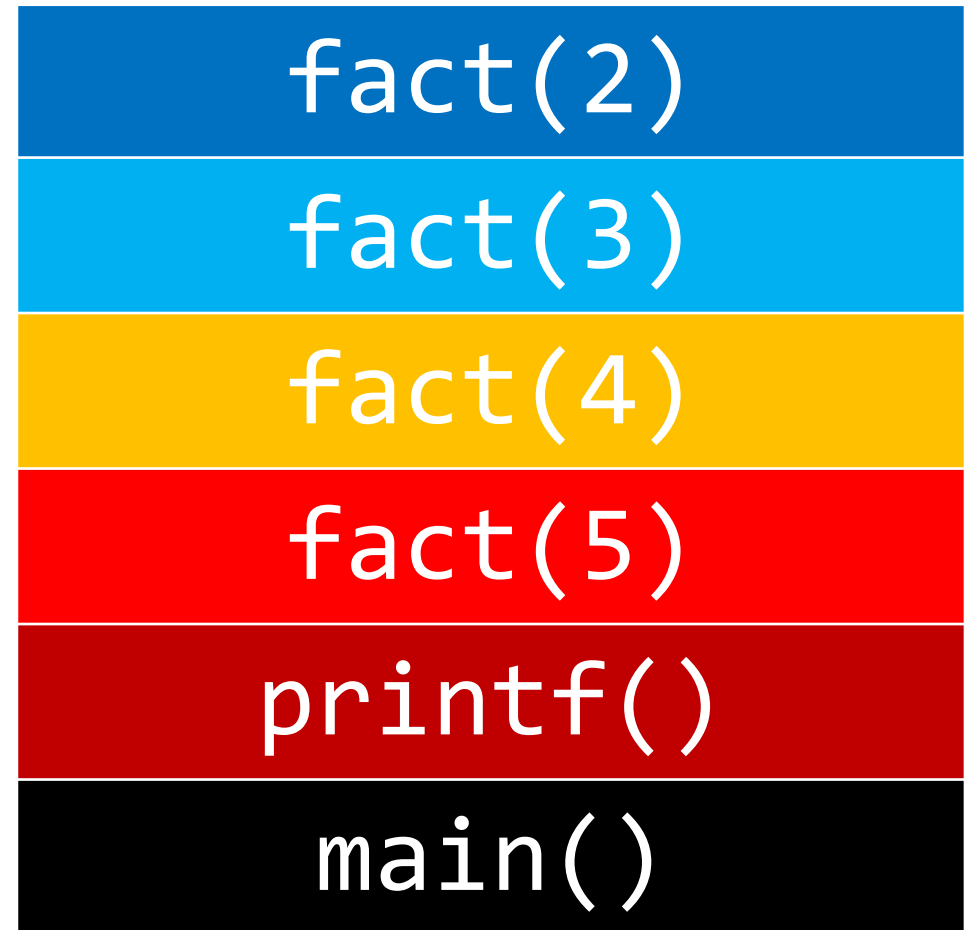


# Call Stack

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



```
int main(void)
{
    printf(“%i\n”, fact(5));
}
```

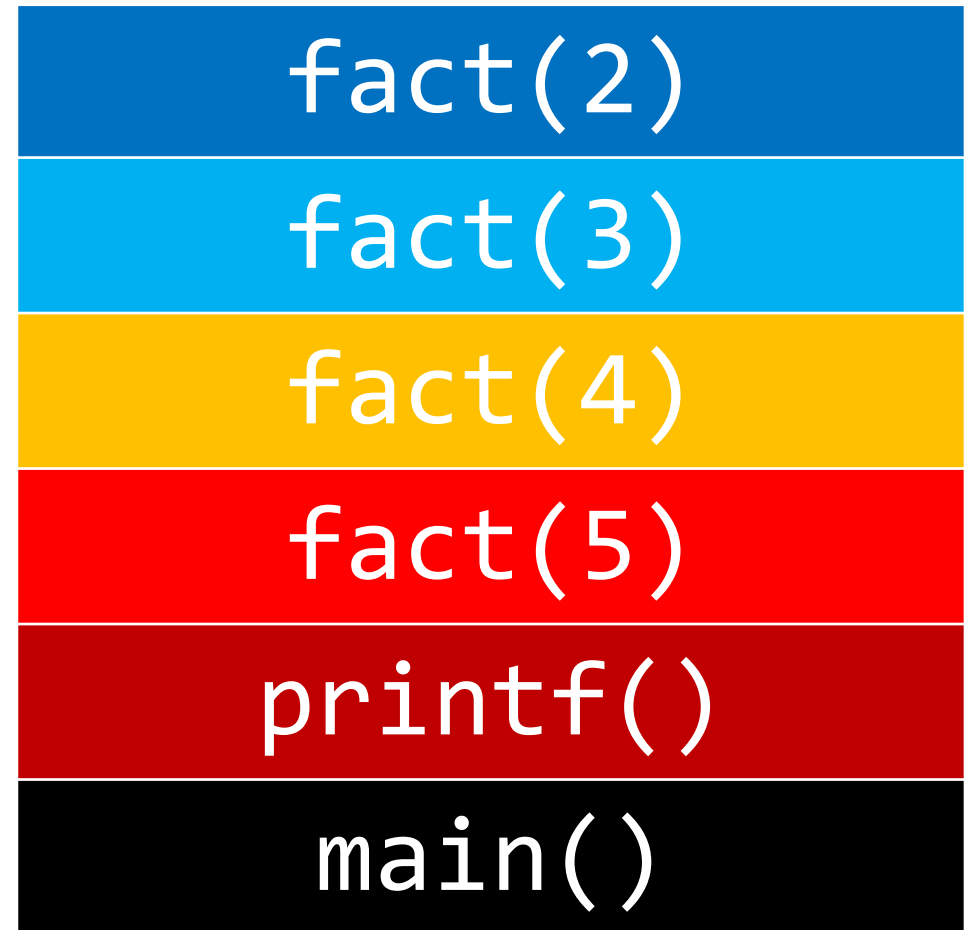




# Call Stack

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

```
int main(void)
{
    printf("%i\n", fact(5));
}
```

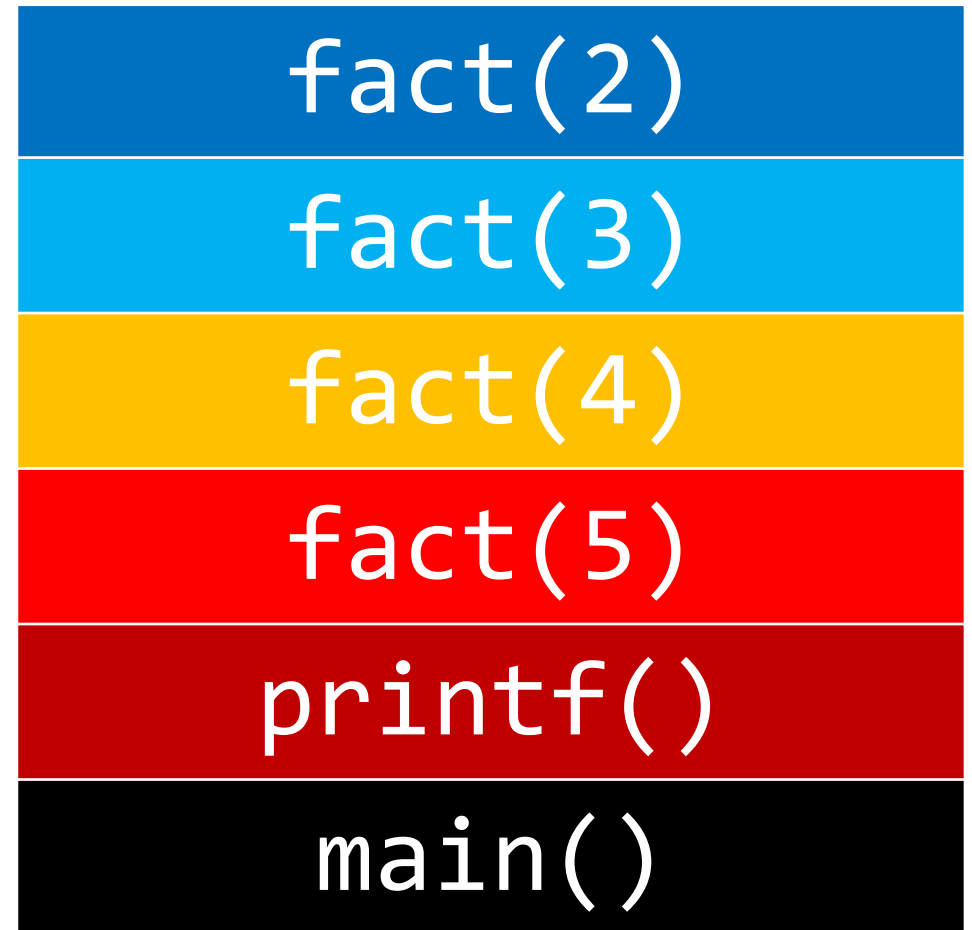


# Call Stack

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



```
int main(void)
{
    printf(“%i\n”, fact(5));
}
```

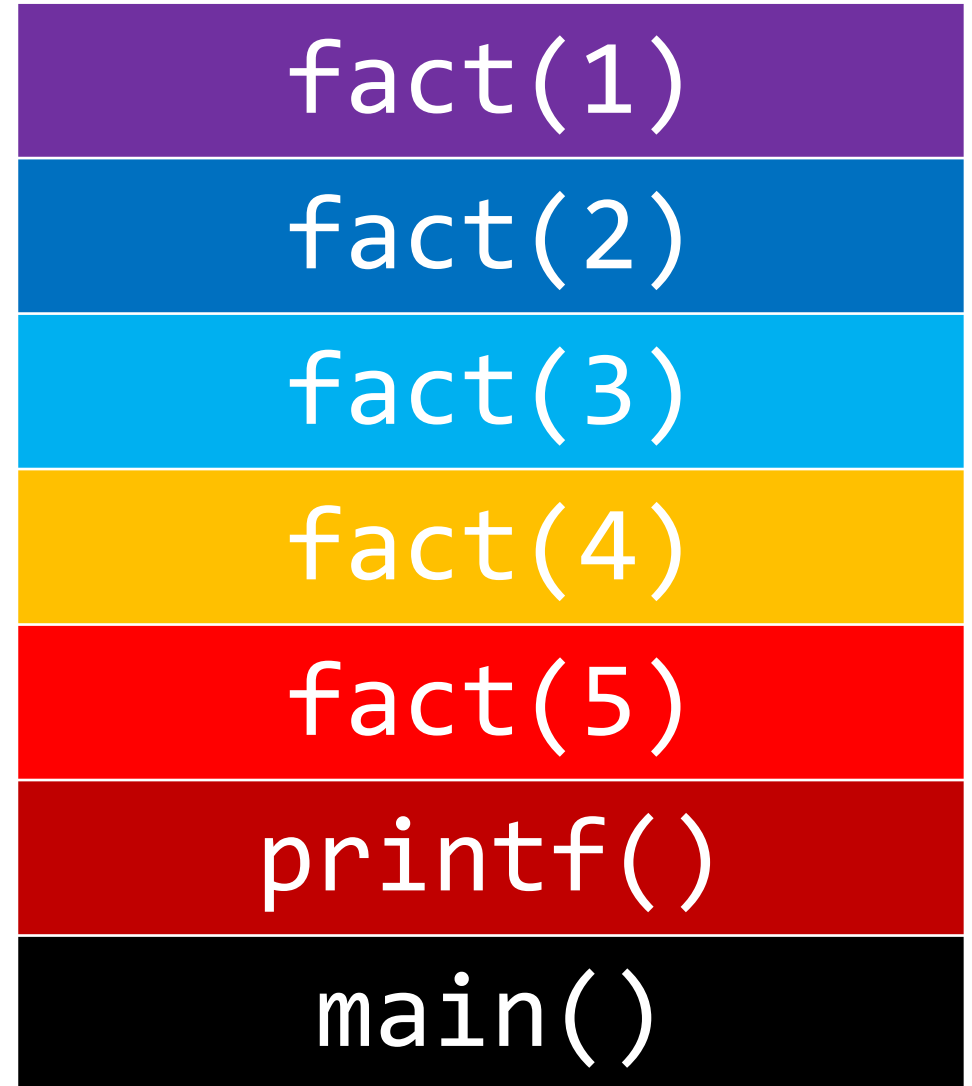


# Call Stack

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



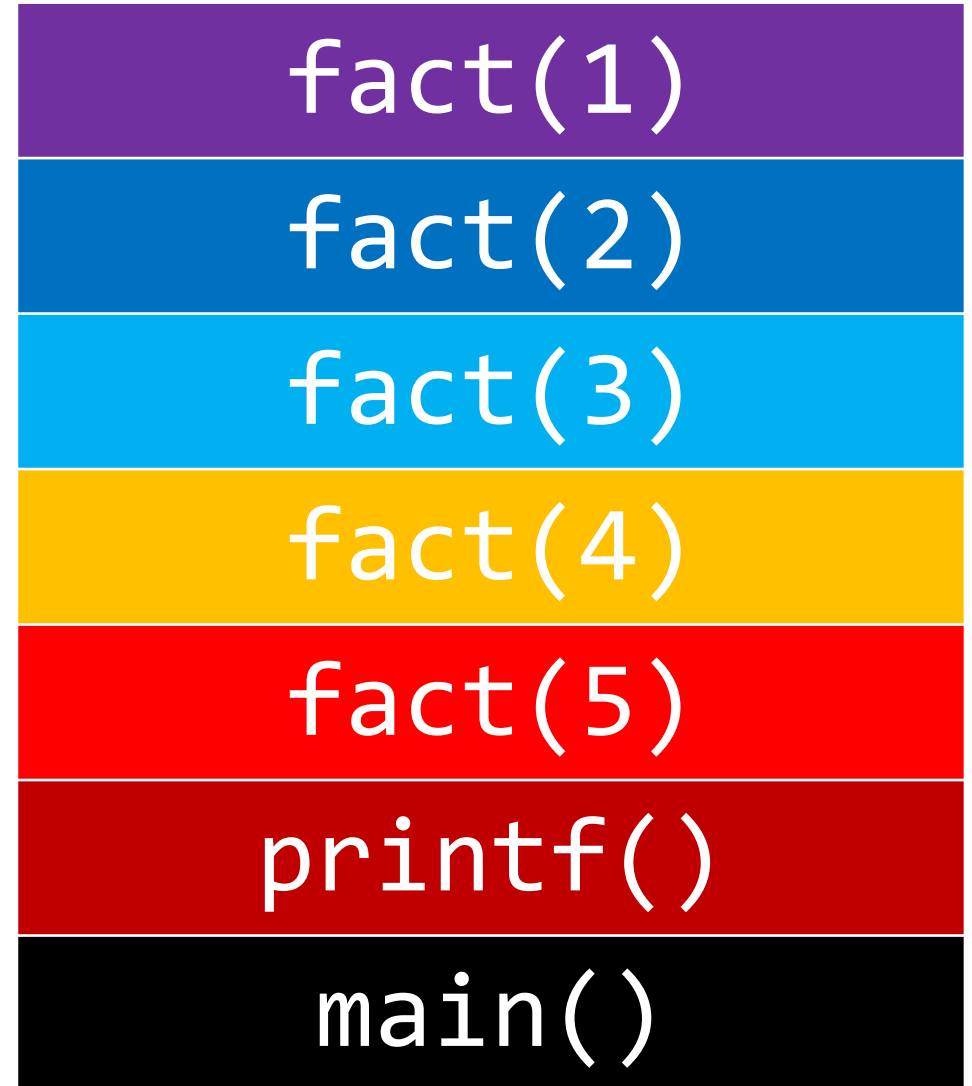
```
int main(void)
{
    printf(“%i\n”, fact(5));
}
```



# Call Stack

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

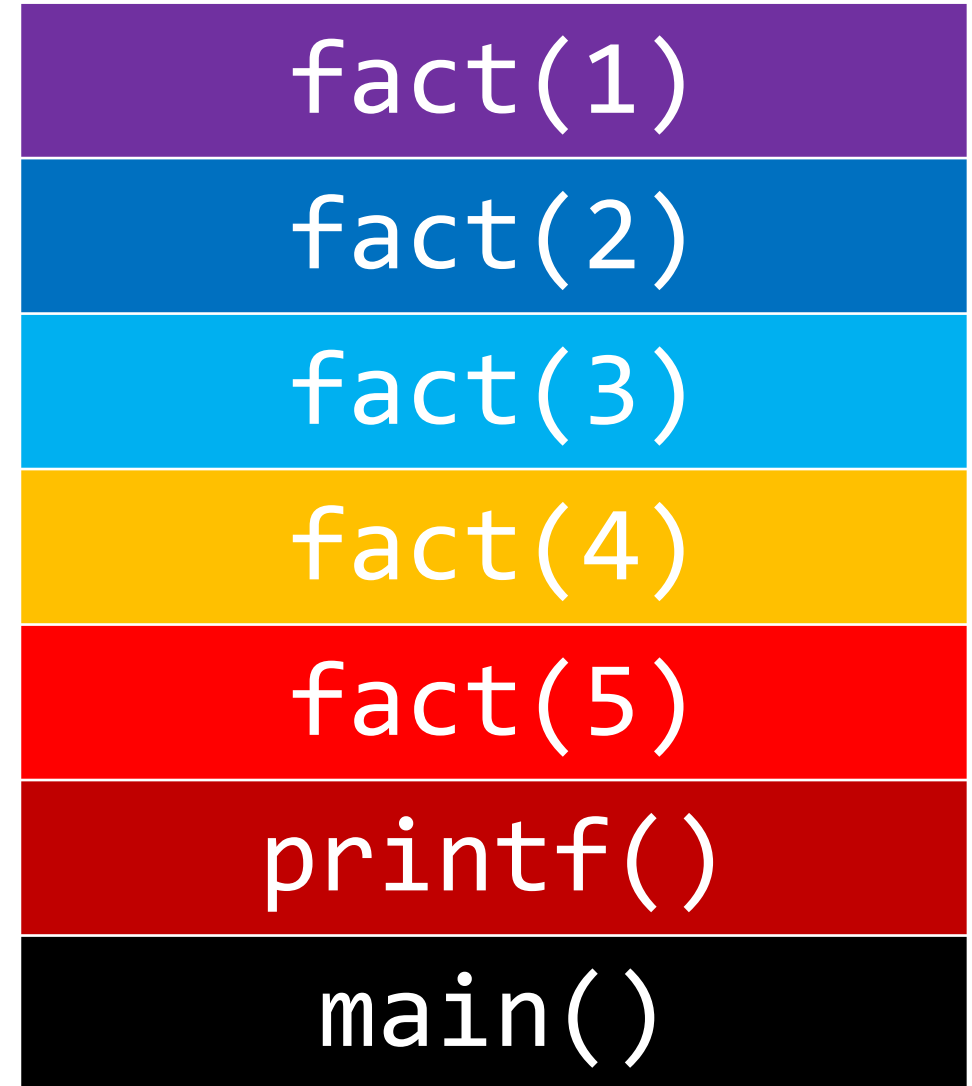
```
int main(void)
{
    printf("%i\n", fact(5));
}
```



# Call Stack

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

```
int main(void)
{
    printf("%i\n", fact(5));
}
```

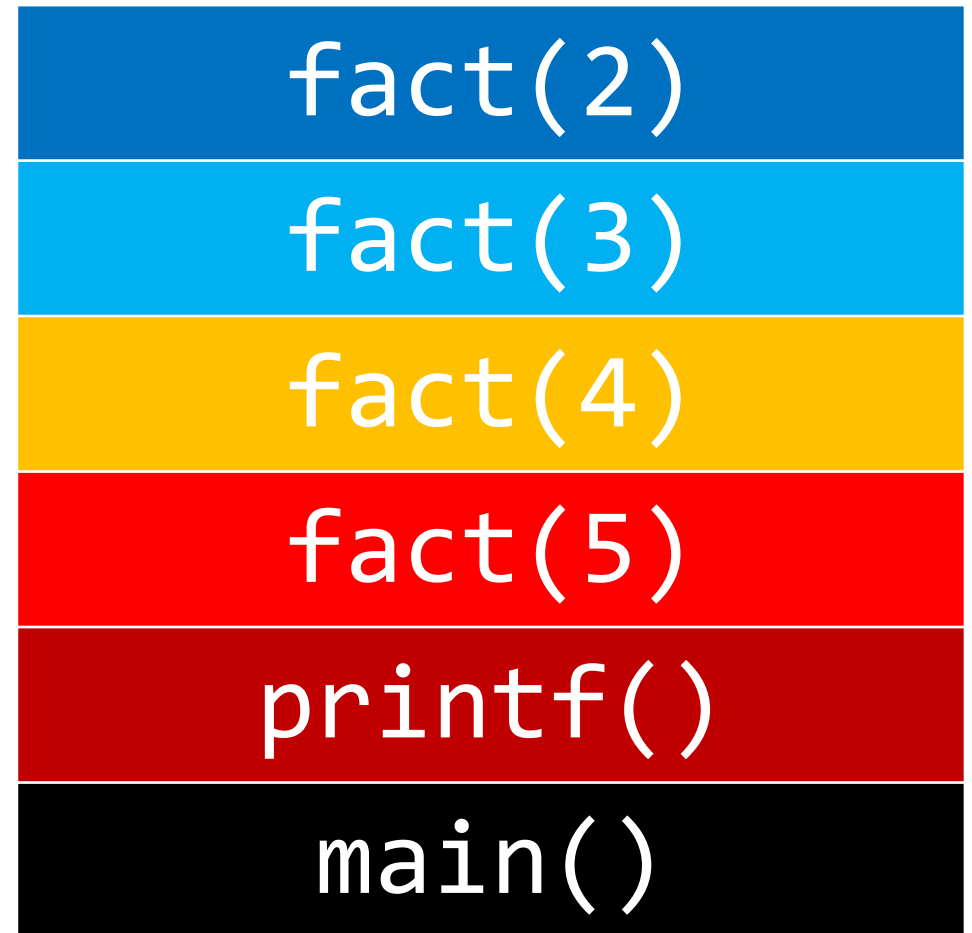


# Call Stack

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



```
int main(void)
{
    printf(“%i\n”, fact(5));
}
```

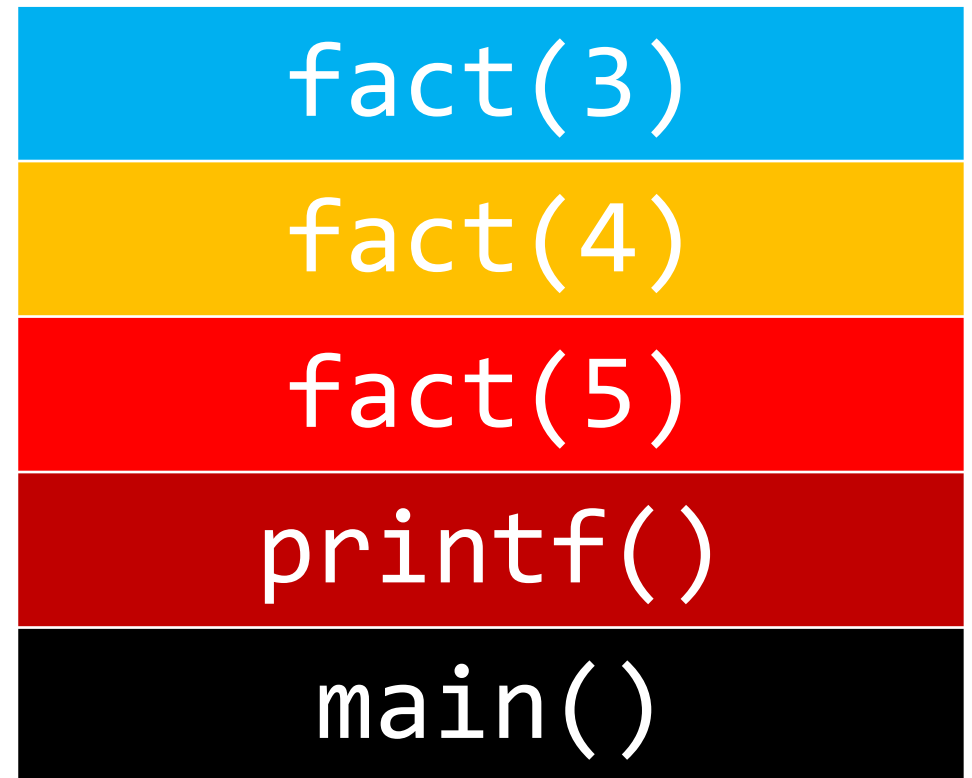


# Call Stack

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



```
int main(void)
{
    printf(“%i\n”, fact(5));
}
```

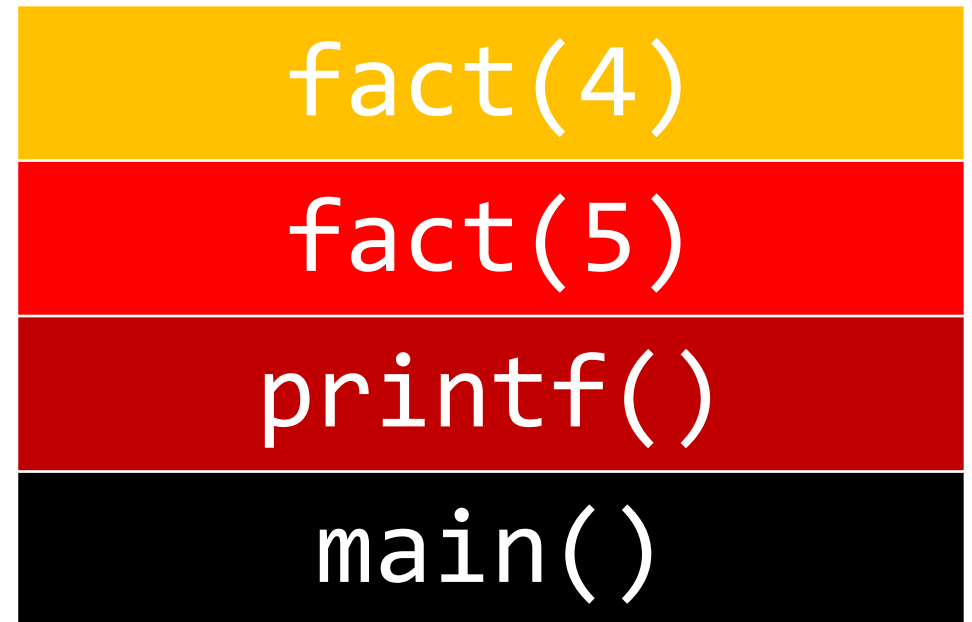


# Call Stack

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



```
int main(void)
{
    printf(“%i\n”, fact(5));
}
```



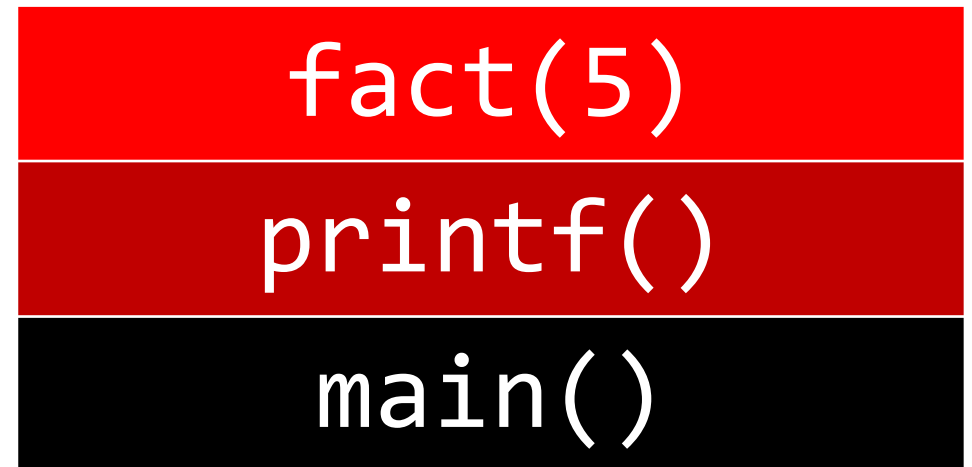


# Call Stack

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



```
int main(void)
{
    printf(“%i\n”, fact(5));
}
```



# Call Stack

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

```
int main(void)
{
    printf(“%i\n”, fact(5));
}
```



`printf()`

`main()`

# Call Stack

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

120

```
int main(void)
{
    printf(“%i\n”, fact(5));
}
```



`printf()`

`main()`

# Call Stack

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

```
int main(void)
{
    printf("%i\n", fact(5));
}
```

main()