Difference between “int main()” and “int main(void)” in C/C++?

Consider the following two definitions of main().

```
int main()
{
   /*  */
   return 0;
}
```

and

```
int main(void)
{
   /*  */
   return 0;
}
```

What is the difference?

In C++, there is no difference, both are same.

Both definitions work in C also, but the second definition with void is considered technically better as it clearly specifies that main can only be called without any parameter.
In C, if a function signature doesn’t specify any argument, it means that the function can be called with any number of parameters or without any parameters. For example, try to compile and run following two C programs (remember to save your files as .c). Note the difference between two signatures of fun().

```
// Program 1 (Compiles and runs fine in C, but not in C++)
void fun() {  }
int main(void)
{
    fun(10, "GfG", "GQ");
    return 0;
}
```

```c
// Program 2 (Fails in compilation in both C and C++)
void fun(void) {  }
int main(void)
{
    fun(10, "GfG", "GQ");
    return 0;
}
```

Unlike C, in C++, both of the above programs fails in compilation. In C++, both fun() and fun(void) are same.

So the difference is, in C, *int main()* can be called with any number of arguments, but *int main(void)* can only be called without any argument. Although it doesn't make any difference most of the times, using "int main(void)" is a recommended practice in C.

Question 1

```c
#include <stdio.h>
int main()
{
    static int i = 5;
    if (--i){
        printf("%d ", i);
        main(10);
    }
}
```

```
4 3 2 1
```

Question 2

```c
#include <stdio.h>
int main(void)
{
    static int i = 5;
    if (--i){
        printf("%d ", i);
        main(10);
    }
}
```

```
prog.c: In function 'main':
prog.c:7:3: error: too many arguments to function 'main'
   main(10);
   ^
```