/* program #1  
  simple program with only a main function */

#include <stdio.h>

int main(void) {
    printf ("Programming is fun. \n");
    return 0;
}

************************************************************************
/* program #2  
  simple program with a main function & one other function */

#include <stdio.h>

int main(void) {
    message();    // function call -- this line calls the other function
    return 0;
}

void message(void) {
    printf ("Programming is fun. \n");
}

************************************************************************

In the second example, there are 2 functions – the main function and another function called message().

All programs have at least a main function – that is always the starting point for any program.

If you copy and paste the program from above with the two functions, and compile it, you will see an error message. Can you guess why???
The reason is that when the compiler is compiling the program, the lines are compiled in the order that they appear in the file, so it will get to the line that says: `message();` before it gets to where that function is defined.

There are two ways to fix this problem, both shown below:

```c
************************************************************************
/* program #3
  simple program with a main function & one other function */
#include <stdio.h>

// solution #1: putting this function before the main
void message(void) {
    printf (“Programming is fun. \n”);
}

int main(void) { // still the starting point when the program is run
    message();
    return 0;
}
************************************************************************
```

```c
************************************************************************
/* program #4
  simple program with a main function & one other function */
#include <stdio.h>

// solution #2: putting this function’s prototype, or signature, before the main
void message(void);

int main(void) {
    message();
    return 0;
}
void message(void); { // prototype for the function
    printf (“Programming is fun. \n”);
}
************************************************************************
```

In program #4 above, the function’s “prototype” is at the top of the program before any other functions (that’s a good place to put all your prototypes). That way, the compiler sees that there is a function called `message()` in the program and when it gets to the function call inside the main function, it is already aware of it.
If a function is returning a value (like how we have been writing `return 0;` at the bottom of our main function) – then we need to change the `void` to the left of the function name to the type of the item that will be returned. Also, you will need a variable in the main to hold the value returned by the function or else the value of the item being returned is lost. Take a look at the following example:

```c
#include <stdio.h>

// prototype
int message(void);

// main function
int main(void) {
    int printMessage;

    printMessage = message();  // assign to the integer variable called printMessage the value returned by the function message() which returns an integer

    if (printMessage == 1)
        printf("That's great! \n");
    else
        printf("That sucks, I'm sorry. \n");

    return 0;
}

// function to return a value of 1 if the user answers the question with a 'Y', or a 0 if the user answers the question with a 'N'
int message(void) {
    char response;
    int yesORno;  // will be given a value below
    // this is the integer item that will be returned from this function
    // back the main function

    printf("Are you having a good day? (Y or N)\n");
    scanf(" %c", &response);

    if (response == 'Y')
        yesORno = 1;
    else
        yesORno = 0;

    return yesORno;
}
```

The above example shows the `message()` function with the word `void` in the parentheses. This means that nothing is being passed to the function; another way of saying it is that the function has no "arguments", or "parameters". The next page shows an example of a function that does have arguments:
program #6
program with a main function & one other function that takes in a value, squares it, and then returns the result */

#include <stdio.h>

int squareTheNumber(int x);

int main(void) {
    int aNumber, theResult;

    printf("Enter an integer: \n");
    scanf("%d", &aNumber);

    theResult = squareTheNumber(aNumber);

    printf("%d squared equals %d \n", aNumber, theResult);

    return 0;
}

int squareTheNumber(int x) {
    int result;

    result = x * x;

    return result;
}