MODULE PRACTICE

Function Prototypes

```
// function prototype
int increment(int);

// function implementation
int increment(int number) {
  return number += 1;
}
```

A function prototype specifies an interface with the required return type and parameter types to help the compiler ensure a function is called properly. A function prototype also helps separate the function declaration from its implementation.

Function Return Value

```
// the return keyword returns the
// value following the keyword
int getOne() {
  return 1;
}
```

A user-defined function can return a value with the `return` keyword followed by the value to be returned. The type of the returned value must match the return type specified in the function signature.

Calling Functions

```c
int incrementBy(int number1, int number2) {
  return number1 + number2;
}

int main() {
  // The value of myNumber is retrieved by
  // calling the function incrementBy() with
  // the arguments 5 and 2
  int myNumber = incrementBy(5, 2);
}
```

In C, a function is called by stating the function name followed by parentheses. One or more argument values can be placed in the parentheses if the function requires any input values.

Function Signature

```c
// A function signature includes the
// return type, function name, and
// parameter(s) in the parentheses
int incrementBy(int number1, number2) {
  return number1 + number2;
}
```

A user-defined function is defined using a function signature. This signature specifies the return type and the function name followed by parameters inside parentheses.

Built-In Function in C

```c
#include <stdio.h>

int main() {
  // printf is a standard library function
  printf("Hello built-in functions!");
}
```

The C programming language comes with built-in standard library functions, such as:

- `printf()`
- `rand()`

Functions in C

A function is a block of reusable logic that may have a defined set of input and output.

Function Parameters

```
// number1 and number2 are paramters
// for the incrementBy function
int incrementBy(int number1, int number2) {
  return number1 + number2;
}
```

In C, a user-defined function can specify input using parameters. Parameters are comma-separated variable definitions within the function signature parentheses.

Return Type `void`

```
// void is used since the function
// printNumber() does not return any value
void printNumber(int number) {
  printf("Your number is %d\n", number);
}
```

A function that returns no value must use the keyword **`void`** as the return type within the function signature.