3 Assignment P1 [Assignment ID: `basics`]

3.1 Preamble (Please Read Carefully)

Before starting work on this assignment, it is **critically important** that you **carefully** read Section 1 (titled "General Information") which starts on page 1 of this document.

3.2 Topics Covered

This assignment covers material primarily related to the following: basic types, references, basic syntax, control flow, looping constructs, const, functions.

3.3 Problems

1. `isPrime` PROGRAM. Write a program called `isPrime` that reads integers from the standard-input stream until end-of-file is reached. The integers being read are to be separated by one or more whitespace characters (e.g., space, tab, newline, carriage return, etc.). For each integer read, the program uses a function called `isPrime` (which returns **`bool`**) to test the integer for primeness. If the integer is prime or not prime, the program writes to the standard-output stream the message "prime" or "composite", respectively, followed by a single newline character. No other output should be generated. The integers read should be greater than one. If this condition is violated, the program should handle this gracefully by simply terminating the program by calling `std::exit(1)` (declared in header file `cstdlib`). Optionally, an error message may also be printed to the standard-error stream (not the standard-output stream). The source code for the `isPrime` program should be placed in the file `isPrime.cpp`.
   Since the integers being read from the standard-input stream are separated by one or more whitespace characters, all of the following text input sequences are equivalent (and should therefore result in identical program output):

   ```
   2 3 4 5
   ```

   ```
   2 3    4
   5
   ```

   ```
   2
   3
   4
   5
   ```

   As an example of correct program behavior, the text input sequence

   ```
   2 3 4 5
   ```

   should produce exactly the following output on the standard-output stream (with the only whitespace characters in the output being newline characters):

   ```
   prime
   prime
   composite
   prime
   ```

   The output generated by the program must follow exactly the specifications given above.
2. `copyInts` FUNCTION. Write a function called `copyInts` that copies the specified number of **`int`** objects from one area of memory to another. You may assume that these two areas do not overlap. The function should take the following parameters (in order):
   (a) a pointer that specifies the first of the **`int`** elements to be copied, where the elements to be copied are contiguous in memory;
   (b) a pointer that specifies the start of the area in memory to which the **`int`** elements are to be copied;
   (c) an unsigned integral type indicating the number of **`int`** elements to be copied.
   The function should not return any value. Also, the function must not use the array subscripting operator (i.e., `[]`). The definition of the function `copyInts` must be placed in a file called `copyInts.cpp`. In addition, a header file called `copyInts.hpp` must be provided that includes a declaration of the function `copyInts`. To further illustrate the behavior of the `copyInts` function, the following code would copy four **`int`** objects from the array `source` to the array `destination`:

```
int source[4] = {1, 2, 3, 4} ; // source for copy
int destination[4]; // destination for copy
copyInts(source, destination, 4); // perform copy
// source and destination arrays now have identical contents
```

Write a program called `testCopyInts` to thoroughly test the `copyInts` function (where the `copyInts` function should be declared in `copyInts.hpp` and defined in `copyInts.cpp`, as explained above). The test code associated with the `testCopyInts` program (including the function `main`) should be placed in a file called `testCopyInts.cpp`.

3. Retrieve the `swap` project in the directory `basics/exercises/swap` from the Git repository for the C++ lecture slides. This directory contains the files:
   - `swap.cpp`. The source code for the `swap` program.
   - `CMakeLists.txt`. A CMakeLists file to build the `swap` program.

   (a) Compile and run the program. Confirm that the program does not work correctly, due to a problem with the `swap` function (which causes it not to perform a swapping operation properly).

   (b) Explain why the `swap` function does not behave correctly, and how this function (and only this function) can be modified in order to obtain the correct behavior.

   No source code needs to be submitted for this problem.