General Sir John Kotelawala Defence University
Department of Computer Science
CS 1011 – Programming Laboratory
Lab Sheet 07

1. A positive integer n is said to be ***prime*** (or, "a prime") if and only if n is *greater than* 1 and is divisible only by 1 and n. For example, the integers 17 and 29 are prime, but 1 and 38 are not prime. Write a function named "`is_prime`" that takes a *positive* integer argument and returns as its value the integer 1 if the argument is prime and returns the integer 0 otherwise. Thus, for example,

```
cout << is_prime(19) << endl; // will print 1
cout << is_prime(1) << endl; // will print 0
cout << is_prime(51) << endl; // will print 0
cout << is_prime(-13) << endl; // will print 0
```

2. Write a function named "`digit_name`" that takes an integer argument in the range from 1 to 9 , inclusive, and prints the English name for that integer on the computer screen. No newline character should be sent to the screen following the digit name. The function should not return a value. The cursor should remain on the same line as the name that has been printed. If the argument is not in the required range, then the function should print "`digit error`" without the quotation marks but followed by the newline character. Thus, for example,

```
digit_name(7); //should print seven on the screen;
digit_name(0); //should print digit error on the screen and place the cursor at the beginning of the next line.
```

3. Write a function named "`reduce`" that takes two positive integer arguments, call them "`num`" and "`denom`", treats them as the numerator and denominator of a fraction, and reduces the fraction. That is to say, each of the two arguments will be modified by dividing it by the greatest common divisor of the two integers.
The function should return the value 0 (to indicate failure to reduce) if either of the two arguments is zero or negative, and should return the value 1 otherwise.

Thus, for example, if `m` and `n` have been declared to be integer variables in a program, then

```
m = 25;
n = 15;
if (reduce(m,n))
     cout << m << '/' << n << endl;
else
     cout << "fraction error" << endl;
```

will produce the following output:

```
5/3
```

Note that the values of m and n were modified by the function

call. Similarly,

```
m = 63;
n = 210;
if (reduce(m,n))
     cout << m << '/' << n << endl;
else
     cout << "fraction error" << endl;
```

will produce the following output:

```
3/10
```

Here is another example.

```
m = 25;
n = 0;
if (reduce(m,n))
     cout << m << '/' << n << endl;
else
     cout << "fraction error" << endl;
```

will produce the following output:

```
fraction error
```

The function `reduce` is allowed to make calls to other functions that you have written.

4. Write a function named "`swap_floats`" that takes two floating point arguments and interchanges the values that are stored in those arguments. The function should return no value. To take an example, if the following code fragment is executed

```
float x =
5.8, y = 0.9;
swap_floats
(x, y);
cout << x << " " << y << endl;
```

then the output will be

```
     0.9 5.8
```

5. Write a function named "`sort3`" that takes three floating point arguments, call them "`x`" , "`y`" , and "`z`", and modifies their values, if necessary, in such a way as to make true the following inequalities: x y z . The function should return no value. To take an example, if the following code fragment is executed

```
float a = 3.2, b =
5.8, c = 0.9; sort3
(a, b, c);
cout << a << " " << b << " " << c << endl;
```

then the output will be

```
0.9 3.2 5.8
```