EECE.2160: ECE Application Programming

Recitation Problems: Functions

The following problems, most of which are taken directly from the textbook, will be covered in recitation during the week of 10/17:

1. Write a function `check_val(x, y, n)` that returns 1 if both `x` and `y` fall between 0 and `n` - 1, inclusive. The function should return 0 otherwise. Assume that `x`, `y`, and `n` are all of type `int`.

2. Write a function `num_digits(n)` that returns the number of digits in `n` (a positive integer). *Hint:* To determine the number of digits in a number *n*, divide it by 10 repeatedly. When *n* reaches 0, the number of divisions indicates how many digits *n* originally had.

3. Write a function `digit(n, k)` that returns the k^{th} digit (from the right) in `n` (a positive integer). For example, `digit(829, 1)` returns 9, `digit(829, 2)` returns 2, and `digit(829, 3)` returns 8. If `k` is greater than the number of digits in `n`, have the function return 0.

4. Write the following function:

   ```
   void split_time(int total_sec, int *hr, int *min, int *sec);
   ```

   `total_sec` is a time represented as the number of seconds since midnight. `hr`, `min`, and `sec` are pointers to variables in which the function will store the equivalent time in hours (0-23), minutes (0-59), and seconds (0-59), respectively.

5. Write the following function:

   ```
   void reduceFraction(int num, int den, int *rNum, int *rDen);
   ```

   This function takes a fraction represented by numerator `num` and denominator `den`, calculates the greatest common divisor (GCD) of those numbers to reduce the fraction, and stores the reduced numerator and denominator in integers pointed to by `rNum` and `rDen`. For example, `reduceFraction(15, 60, &r, &d)` would reduce the fraction 15/60 to 1/4, storing 1 in `r` and 4 in `d`. Assume the denominator is always non-zero.

   The algorithm for finding the GCD of two numbers, *x* and *y*, is as follows:

   a. If *y* is 0, *x* is the GCD.

   b. Otherwise, calculate *r*, the remainder of *x* / *y*.

   c. Let *x* = *y*, and *y* = *r*. (In other words, *x* holds the “old” value of *y*, and the new value of *y* is the remainder from Step b)

   d. Return to Step a.