

CSE1301 Exercise Sheet 4 Functions

Exercise 1

We are going to write some algorithms as functions (though will not code them just yet)

(a) Write an algorithm *isLeapYear* as a function that determines whether a given year is a leap year. Pass the year as a parameter. A year is a leap year if

- It is a multiple of 4 but not a multiple of 100

OR

- It is a multiple of 400

So, for example, 1996 and 2000 are leap years, but 1900, 2002 and 2100 are not.

(b) Write an algorithm *daysInMonth* as a function. It should determine how many days in the month there are for a given year. Pass the month and the year as parameters. This function should call the *isLeapYear* function.

(c) Write a function to compute number of days between any two dates. This function should call the functions *DaysInMonth* and *isLeapYear*.

(d) Write a function for determining the day of the week on which your birthday will fall in any year, given today's date and the date on which you were born. You will probably find it easiest if you call a function you have already written. What do you have to pass to your function? What will your function return?

(e) When you come to code these functions in C, what will be the types of the various parameters? What will be the return types?

Exercise 2 (from 1995 Sample Exam)

Write a C function which takes an integer parameter (greater than 1), and returns the highest power of that integer which is also less than 1,000.

For example:

Given the parameter value 2 your function should return 512 (2^9)

Given the parameter value 3 your function should return 729 (3^6)

Given the parameter value 7 your function should return 343 (7^3)

Given the parameter value 42 your function should return 42 (42^1)

Exercise 3

On a computer screen, the *pixels* (points) are given nonnegative integer coordinates. The upper-left corner pixel is (0,0). The horizontal coordinate *x* increases as you go from left to right. The vertical coordinate *y* increases as you go from the top to the bottom. (This is upside-down from the Cartesian coordinate system you learned in geometry.)

(a) Declare global constants *WIDTH* and *HEIGHT*, to represent the maximum number of pixels in each direction on your screen, and initialise them appropriately.

(b) Write C functions to do the following. Apart from the above global constants, all information to be passed from one function to another must be passed as parameters.

- (i) Return the maximum of two given integers.
- (ii) Return the minimum of two given integers.

- (iii) Take a pair of integers (x,y) and determine if it represents a legal pixel (i.e., a pixel on the screen) or not.
- (iv) Take a point (x,y) , which may or may not fall within the bounds of the screen, and find the nearest pixel on the screen to that point. This function should use your previously written functions, as much as possible.
- (v) Take a pixel (x,y) and add a vector (a,b) to it, giving $(x+a,y+b)$ if the result is on the screen, or the nearest point on the edge of the screen otherwise. This function should use your previously written ones, as much as possible.

Exercise 4

Write C prototypes for functions to do the following

- (a) calculate the cube root of a number of type double;
- (b) find a cricketer's batting average, given total runs scored and number of times out;
- (c) find the term in years of a bank loan, given the amount of the loan, the frequency of repayments, the desired maximum amount of each repayment, and the interest rate;
- (d) decide whether or not a character is alphabetic;
- (e) print the average of three integer parameters, but don't return it;
- (f) read in a float as input from the user, and return its integer part;
- (g) print a table of ASCII values of all letters in the alphabet.

ADVANCED EXERCISES

Extend the set of functions that you created in Exercise 3 to do rotation of points by various angles about (a) the origin, or (b) another point. Start with easy angles like 90 degrees or 180 degrees. Then try to deal with general angles.

Then deal with reflection about a line through the origin, again starting with easy cases first, then the general case.

Other operations may also come to mind.