PSPD Assignment - 05

4. Write a program to take two numerical lists of the same length ended by a sentinel value and store the lists in arrays `x` and `y`, each of which has 20 elements. Let `n` be the actual number of data values in each list. Store the product of corresponding elements of `x` and `y` in a third array, `z`, also of size 20. Display the arrays `x`, `y`, and `z` in a three-column table. Then compute and display the square root of the sum of the items in `z`. Make up your own data, and be sure to test your program on at least one data set with number lists of exactly 20 items. One data set should have lists of 21 numbers, and one set should have significantly shorter lists.

5. A barcode scanner for Universal Product Codes (UPCs) verifies the 12-digit code scanned by comparing the code's last digit (called a *check digit*) to its own computation of the check digit from the first 11 digits as follows:
   1. Calculate the sum of the digits in the odd-numbered positions (the first, third, ..., eleventh digits) and multiply this sum by 3.
   2. Calculate the sum of the digits in the even-numbered positions (the second, fourth, ..., tenth digits) and add this to the previous result.
   3. If the last digit of the result from step 2 is 0, then 0 is the check digit. Otherwise, subtract the last digit from 10 to calculate the check digit.
   4. If the check digit matches the final digit of the 12-digit UPC, the UPC is assumed correct.

   Write a program that prompts the user to enter the 12 digits of a barcode separated by spaces. The program should store the digits in an integer array, calculate the check digit, and compare it to the final barcode digit. If the digits match, output the barcode with the message "validated." If not, output the barcode with the message "error in barcode." Also, output with labels the results from steps 1 and 2 of the check-digit calculations. Note that the "first" digit of the barcode will be stored in element 0 of the array. Try your program on the following barcodes, three of which are valid. For the first barcode, the result from step 2 is 79 (0 + 9 + 0 + 8 + 4 + 0) * 3 + (7 + 4 + 0 + 0 + 5).

   0 7 9 4 0 0 8 0 4 5 0 1
   0 2 4 0 0 0 1 6 2 8 6 0
   0 1 1 1 1 0 8 5 6 8 0 7
   0 5 1 0 0 0 1 3 8 1 0 1

7. A normalized vector X is defined as

$$x_i = \frac{v_i}{\sqrt{\sum_{i=1}^{n} v_i^2}}; \qquad i = 1, 2, \ldots, n$$

   Each element of the normalized vector X is computed by dividing the corresponding element (v_i) of the original vector by the square root of the sum of the squares of all the original vector's elements. Design and test a program that repeatedly scans and normalizes vectors of different lengths. Define functions `scan_vector`, `normalize_vector`, and `print_vector`.

12. The *binary search* algorithm that follows may be used to search an array when the elements are in order. This algorithm is analogous to the following approach for finding a name in a telephone book.
    a. Open the book in the middle, and look at the middle name on the page.
    b. If the middle name isn't the one you're looking for, decide whether it comes before or after the name you want.
    c. Take the appropriate half of the section of the book you were looking in and repeat these steps until you land on the name.

    Write and test a function `binary_srch` that implements this algorithm for an array of integers. When there is a large number of array elements, which function do you think is faster: `binary_srch` or the linear search function of Fig. 7.14?

13. The *bubble sort* is another technique for sorting an array. A bubble sort compares adjacent array elements and exchanges their values if they are out of order. In this way, the smaller values "bubble" to the top of the array (toward element 0), while the larger values sink to the bottom of the array. After the first pass of a bubble sort, the last array element is in the correct position; after the second pass the last two elements are correct, and so on. Thus, after each pass, the unsorted portion of the array contains one less element. Write and test a function that implements this sorting method.

14. A C program can represent a real polynomial $p(x)$ of degree n as an array of the real coefficients $a_0, a_1, \ldots, a_n$ $(a_n \neq 0)$.

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

    Write a program that inputs a polynomial of maximum degree 8 and then evaluates the polynomial at various values of x. Include a function `get_poly` that fills the array of coefficients and sets the degree of the polynomial, and a function `eval_poly` that evaluates a polynomial at a given value of x. Use these function prototypes:

```
void get_poly( double coeff[], int* degreep );
double eval_poly( const double coeff[], int degree,
                  double x );
```