LAB ASSIGNMENTS

Problem Solving and Program Design Using C

(CSE 3942)

Department of Computer Science & Information Technology
Faculty of Engineering & Technology (ITER)
Siksha 'O' Anusandhan Deemed To Be University
Bhubaneswar, Odisha - 751030

Lab Assignment-6

6. Programming project on Arrays and Strings

6.1 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.

6.2 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).

079400804501

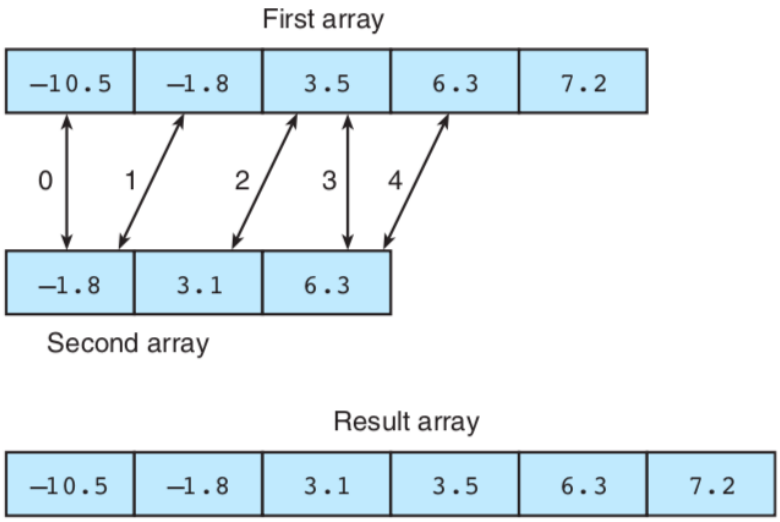
024000162860

011110856807

051000138101

6.3 Each year the Department of Traffic Accidents receives accident count reports from a number of cities and towns across the country. To summarize these reports, the department provides a frequency distribution printout that gives the number of cities reporting accident counts in the following ranges: 0–99, 100–199, 200–299, 300–399, 400–499, and 500 or above. The department needs a computer program to take the number of accidents for each reporting city or town and add one to the count for the appropriate accident range. After all the data have been processed, the resulting frequency counts are to be displayed.

6.4 Write a function that will merge the contents of two sorted (ascending order) arrays of type double values, storing the result in an array output parameter (still in ascending order). The function should not assume that both its input parameter arrays are the same length but can assume that one array does not contain two copies of the same value. The result array should also contain no duplicate values.



Hint: When one of the input arrays has been exhausted, do not forget to copy the remaining data in the other array into the result array. Test your function with cases in which (1) the first array is exhausted first, (2) the second array is exhausted first, and (3) the two arrays are exhausted at the same time (i.e., they end with the same value). Remember that the arrays input to this function must already be sorted.

6.5 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.

ALGORITHM FOR BINARY SEARCH

1. Let **bottom** be the subscript of the initial array element.
2. Let **top** be the subscript of the last array element.
3. Let **found** be false.
4. Repeat as long as **bottom** isn't greater than **top** and the target has not been found
5. Let middle be the subscript of the element halfway between **bottom** and **top**.
6. if the element at middle is the target
7. Set **found** to true and **index** to middle.
   else if the element at middle is larger than the target
8. Let **top** be middle - 1.
   else
9. Let **bottom** be middle + 1.

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?

6.6 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.

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

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

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);

6.8 Write and test a function **deblank** that takes a string output and a string input argument and returns a copy of the input argument with all blanks removed.

6.9 Write and test a function **hydroxide** that returns a 1 for true if its string argument ends in the substring **OH**.

Try the function hydroxide on the following data:

KOH H2O2 NaCl NaOH C9H8O4 MgOH

6.10 Write a program that takes nouns and forms their plurals on the basis of these rules:

a. If noun ends in "y", remove the "y" and add "ies".
b. If noun ends in "s", "ch", or "sh", add "es".
c. In all other cases, just add "s".

Print each noun and its plural. Try the following data:

chair dairy boss circus fly dog church clue dish

6.11 Write a program that takes data a line at a time and reverses the words of the line. For example,

Input: birds and bees

Reversed: bees and birds

The data should have one blank between each pair of words.

6.12 Write and test a function that finds and returns through an output parameter the longest common suffix of two words (e.g., the longest common suffix of "procrastination" and "destination" is "stination", of "globally" and "internally" is "ally", and of "gloves" and "dove" is the empty string).