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-5

5. Programming project on Pointers and Modular Programming

5.1 Write a program for an automatic teller machine that dispenses money. The user should enter the amount desired (a multiple of 10 dollars) and the machine dispenses this amount using the least number of bills. The bills dispensed are 50s, 20s, and 10s. Write a function that determines how many of each kind of bill to dispense.

5.2 Write a program to dispense change. The user enters the amount paid and the amount due. The program determines how many dollars, quarters, dimes, nickels, and pennies should be given as change. Write a function with four output parameters that determines the quantity of each kind of coin.

5.3 Determine the following information about each value in a list of positive integers.

a. Is the value a multiple of 7, 11, or 13?

b. Is the sum of the digits odd or even?

c. Is the value a prime number?

You should write a function with three type int output parameters that send back the answers to these three questions. Some sample input data might be:
104 3773 13 121 77 30751

5.4 The square root of a number N can be approximated by repeated calculation using the formula

$$NG = 0.5(LG + N/LG)$$

where NG stands for next guess and LG stands for last guess. Write a function that calculates the square root of a number using this method.

The initial guess will be the starting value of LG . The program will compute a value for NG using the formula given. The difference between NG and LG is checked to see whether these two guesses are almost identical. If they are, NG is accepted as the square root; otherwise, the next guess (NG) becomes the last guess (LG) and the process is repeated (another value is computed for NG, the difference is checked, and so on). The loop should be repeated until the difference is less than 0.005. Use an initial guess of 1.0.

Write a driver function and test your square root function for the numbers 4, 120.5, 88, 36.01, 10,000, and 0.25.

5.5 When an aircraft or an automobile is moving through the atmosphere, it must overcome a force called drag that works against the motion of the vehicle. The drag force can be expressed as

$$F = \frac{1}{2} CD \times A \times \rho \times V^2$$

where F is the force (in newtons), CD is the drag coefficient, A is the projected area of the vehicle perpendicular to the velocity vector (in m^2), is the density of the gas or fluid through which the body is traveling (kg/m^3), and V is the body's velocity. The drag coefficient CD has a complex derivation and is frequently an empirical quantity. Sometimes the drag coefficient has its own dependencies on velocities: For an automobile, the range is from approximately 0.2 (for a very streamlined vehicle) through about 0.5. For simplicity, assume a streamlined passenger vehicle is moving through air at sea level (where $\rho = 1.23 kg/m^3$). Write a program that allows a user to input A and CD interactively and calls a function to compute and return the drag force. Your program should call the drag force function repeatedly and display a table showing the drag force for the input shape for a range of velocities from 0 m/s to 40 m/s in increments of 5 m/s.

5.6 Write a program to model a simple calculator. Each data line should consist of the next operation to be performed from the list below and the right operand. Assume the left operand is the accumulator value (initial value of 0). You need a function **scan_data** with two output parameters that returns the operator and right operand scanned from a data line. You need a function **do_next_op** that performs the required operation. **do_next_op** has two input parameters (the operator and operand) and one input/output parameter (the accumulator). The valid operators are:

+ add
− subtract
∗ multiply
/ divide
ˆ power (raise left operand to power of right operand)
q quit

Your calculator should display the accumulator value after each operation. A sample run follows.

```
+ 5.0
result so far is 5.0
ˆ 2
result so far is 25.0
/ 2.0
result so far is 12.5
q   0
final result is 12.5
```

5.7 After studying gross annual revenues of Broadway shows over a 20-year period, you model the revenue as a function of time:

$$R(t) = 203.265 \times (1.071)^t$$

where R is in millions of dollars and t is the years since 1984. Create the following C functions to implement this model:

revenue—calculates and returns R for an input parameter of t.

predict—predicts the year in which revenues (in millions) will first equal or exceed the value of the input parameter. For example, **predict(200)** would return 1984.

Write a main function that calls **predict** to determine when revenues will likely exceed $1 trillion (i.e., 1,000 million). Then create an output file that contains a table of estimated revenues (in millions of dollars) for all the years from 1984 through the year when revenues should exceed $1 trillion. Round revenue estimates to three decimal places.

5.8 Since communications channels are often noisy, numerous ways have been devised to ensure reliable data transmission. One successful method uses a checksum. A checksum for a message can be computed by summing the integer codes of the characters in the message and finding the remainder of this sum divided by 64. The integer code for a space character is added to this result to obtain the checksum. Since this value is within the range of the displayable characters, it is displayed as a character as well. Write a program that accepts single-line messages ending with a period and displays the checksum character for each message. Your program should continue displaying check- sums until the user enters a line with only a period.

5.9 Harlan A. Brothers and John A. Knox discovered that as the value of x gets larger, the value of the expression $\left(\frac{2x+1}{2x-1}\right)^x$ gets closer and closer to e. Write a program that evaluates this expression for $x = 1, 2, 3$, and so on until the absolute difference between the expression's value and the value of e calculated by the library function **exp** is less than 0.000001. Display the value of x that causes your loop to exit along with both the final approximation of e and the value of e calculated by the **exp** function. Show 7 decimal places.