Assignment 1

-Taneesha Gaur

REPORT

Problem statement – Design and implement a C library for integers of arbitrary length ("intal" in short for integers of arbitrary length).

Assumption

The length of input taken by the code is limited by the range of malloc function (used in the code).

The range of the exponent in the power function is limited by the range of the size of the integer.

The user enters valid inputs/integers , such as 1233 , rather than 1+23 or 1sd32 , which aren't integers .

The user enters new values into the intal_demo.c file , and compiles and runs the file for those inputs .

Changes made in the original template – there was one change which I made in the read_intal functin that had been given by channa sir . In that , he hadn't assigned the value of strlen(a->s)-1 back to a->n . also , I had to make changes to the read_intal function , as in the struct definition , it had specified that the sign should not be included in a->n , but it was included in the read_intal function .

Approach

Most of the functions have been implemented using brute force techniques. Some functions use other functions calls in them so as to make the implementation easier , such as the division and power function , which use subtraction and multiplication respectively

Addition function

I have implemented addition function by using brute force algorithm for adding two array of integers . The last two integers in the arrays (or chars) are added and sum is stored in a temporary variable . The value of the variable shouldnt be more than 10 , and if it is the carry should be included in the addition for the next two integers of both the arrays . When we try to add opposite signed integers with my code , it

redirects the inputs to the subtraction function , rather than doing subtraction in addition itself.

Subtraction function

Have implemented the subtract_intal function such that the larger number is always the one being subtracted from(first number a->s is larger than b->s) , and the signs of the subtraction are also handled appropriately . The same brute force algorithim is used for subtraction too , but here we are calculating the difference between the last element of the first array and the second array , and checking whether that value is greater than 0 . If lesser , then we are borrowing one from the previous a->s[i]th element , and adding 10 to the current difference . The problem with different signed integers to be subtracted is also taken care of (using add_intal function call).

Multiplication function

implemented brute force multiplication algorithm for this function . the last digit of the multiplier array is multiplied once with the multiplicand and the value is stored in an array . Then , the product of the next multiplier digit is calculated and stored from i'th to the nth digit in the array , and if some value is alread present in the array at thta position , add that to the existing value. Finally converted the integer array to a character array , keeping in mind that no digit can exceed value more than 10.

Division function

Repeated subtraction of the numerator with the denominator , and a track of the quotient is kept . While the value of the numerator is greater than or equal to the denominator , keep subtracting numerator and increment the quotient . However , if the division function is tried with larger digit numbers , in which many function calls have to be made , then the code crashes as stack overflow occurs .

Exponentiation function

Repeated multication of the base with itself , and keep decrementing the power or the exponent until it becomes 1. Then return the finalvalue through an intal .

Learning

I have learnt how to reduce the length of the code once a brute force approach has been implemented , and also how to debug the large program . Also , to avoid memory leaks in code , one should keep freeing the pointers and arrays which the programmer has allocated.And by discussing the assignment with my classmates and friends , I have been able realize the different ways to solve this assignment , and also have tried implementing the method I liked best.