Develop a C library for the integers of arbitrary length (intal)

Name: Ravichandra G
SRN: PES1201701581

Introduction:

An intal is a nonnegative integer of arbitrary length represented as a string of decimal digits ('0' thru '9') that are stored in the big-endian style. The integer is stored as a null-terminated string of ASCII characters. The most significant digit in intal is at the head of the string and it terminates with null character.

C numerical data types support a maximum 2^64 bit number which is in the range of -2^63 to 2^63-1 or 0 to 2^64-1.Intal is a nonnegative integer of arbitrary length represented as a string of decimal digits. Intal has no limit. Integer data type has size limits so we can not perform some basic numerical operations like addition, subtraction, multiplication, power of two large numbers, there might be overflow because of size limit. Using Intal library we can overcome these problems. Intal supports all types of operations without any overflow.

All numerical operations can be performed by using intals efficiently. It is possible to store very large integers using intals.

Approach:

While performing basic arithmetic operations like addition, subtraction, multiplication on intals we need to convert char type to int type on which actual operations are performed by taking care of carry and borrow then again resultant integer is converted to char to store it in resultant intal. Intal_mod, intal_pow, intal_gcd, intal_fibonacci, intal_factorial, intal_bincoeff, intal_binarysearch, intal_coinrowproblems all are implemented using basic arithmetic operations intal_addition, intal_substraction, intal_multiplication and intal_compare function. I have used some inbuilt functions like strlen(), strcpy(), strdup() etc of string library to deal with strings.

FUNCTIONS:

1.Addition of two intals:

Make intal2 as the larger one by swapping if necessary, and add two digits at a time.Store carry and update the value everytime into the final result until the length of the smaller number that is intal1 is reached. Later keep that carry and iterate through the intal2 and perform the same addition. We set the first bit of the final result to ‘1’ when there’s overflow.

2.Comparing two intals:

First we check the length of the two intals, the intal with the larger length is bigger. Next we use the strcmp function of the standard library to compare the strings. It gives 0 if both are same, positive number if intal1 is greater than intal2. With the help of this we’ll return 1 if intal1 is bigger or else -1.

3. Difference of two intals:

Make swaps if necessary to make intal1>intal2 so that difference is always nonnegative.
Subtraction of two intals has following steps

a. Reverse both strings.
b. Keep subtracting digits one by one from 0’th index (in reversed strings) to end of smaller string, append the diff if it’s positive to end of result. If difference(diff) is negative, then add 10 and keep track of carry as 1 if it’s positive then carry is 0.
c. Finally reverse the result.

4. Multiplication of two intals:

Multiplication of two intals is similar to multiplication of integers.
First reverse the intals. we traverse intal1 and multiply each digit of intal1 with each digit of intal2.All variables carry,quotient ,remider are updated accordingly and the intermediate result is stored, which will be used later. Finally the trailing zeros are removed from the result and the result is reversed.

5. Smallest intal in the array of intals:

While finding the smallest intal, we use the function intal_compare as the helper function. With the help of this we use the traditional logic to find the smallest number ie initialize first intal of the array as smallest and call intal_compare with every other intal, update the minimum and index if found and return the index.

6. Largest intal in the array of intals:

While finding the largest intal, we use the function intal_compare as the helper function. With the help of this we use the traditional logic to find the largest number i.e initialize first intal of the arry as max and call intal_compare with every other intal, update the max and index if found and
return the index.

7. Intal search => Linear Search:

With the help of the helper function intal_compare, we iterate through the intal array and call intal_compare with parameters intal and the key. If it returns 0 then we return the index corresponding to the match.

8. Intal binary search

Binary search algorithm is applied to the intals by calling intal_compare function when there is comparison required between the intal and the key. If the intal is found we return the index(mid) or else -1 when the intal is not found in the array.

9. Intal Fibonacci:

The result of nth fibonacci element can be very large. So we store the result as intal. Initially we set up two strings, 0,1. With the help of recurrence relation f(n) = f(n-1)+f(n-2). We
generate the fibonacci number. The addition involved is performed by the function intal_add which was implemented earlier.

10. Intal Factorial :

Initially we'll have two temp var with value 1 in the form of string. We iteratively call the intal_multiply and intal_add with parameters temp1 and temp2 where we increment temp2 after every iteration to calculate the factorial of the given number. Number of iterations is the value 'n' passed to the intal_fibonacci.

11.Sort an array of intals:

We implemented the traditional quick sort method for sorting an array of intals except the fact that we use Intal_compare for comparing two intals.

12.Mod of two intals:

Mod of two intals are calculated by iteratively finding the difference between intal1 and intal2, updating the difference(intal_difference) in intal1 and iteratively finding the product(intal_multiply) of the two intals and updating it into intal2. This loops ends when intal1's
value is greater than intal2 and the value in intal1 is returned. In the case of both being the same, return 0.

13. GCD of two intals:

GCD of two intals are calculated using the Euclid's Algorithm where for comparison and mod we use intal_compare and inal_mod functions.

14.power(intal,int n):

Power of the intal is calculated by recursively multiplying with the intal with itself if n is even.

This is done in O(logn) i.e. by calling the intal_pow function recursively. If n is odd then we multiply the resultant intal instead of the intal.

15.intal_bincoeff(unsigned int n, unsigned int k):

Binomial coefficient of an integer can be very large. So we store the result of binomial as intal.we make use of the Pascal's identity C(n,k) = C(n-1,k) + C(n-1,k-1)
We use dynamic programming to find the C(n,k).
It’s recursion follows like this C(n,k) = C(n-1,k) + C(n-1,k-1) , C(n, 0) = C(n, n) = 1
We use intal_add method for addition.

16. Coin_row_problem for array of intals:

Coin_row_problem finds the maximum sum such that no two elements are consecutive.
Initialize the incl variable to the first element of the array. Initialize excl to 0.
Traverse array and from second element and store the maximum of incl and excl in excl_new ,assign incl tp exl+arr[i] and store excl_new in excl
Finally return the maximum of incl and excl