**General Info**

This project is a simple C library ‘intal’ short for integer of arbitrary length, a library of nonnegative integers of arbitrary length.

**Technologies**

The language used for the implementation of intal is C. All code is written in C language.

The main builtin library that are used:

* Stdio.h
* String.h
* Stdlib.h

PES1201701633.c file implements all the functionalities declared in the header file ‘intal.h’. Client files for the above library are intal\_sanity\_check.c and intal\_client\_sample.c

**Setup**

To run this project execute following instructions in the terminal

gcc -Wall PES1201701633.c intal\_client\_sample.c -o output.out

./output.out

**INTRODUCTION**

Library “intal” short for integer of arbitrary length , a library of nonnegative integers of arbitrary length. The given header file “intal.h” declares the functionalities the library is expected to provide. The implementation file, declares the structure of the intal along with defining the functionalities declared in intal.h.

Client treates an intal (integer of arbitrary length) as an object pointed by a pointer “void”. An intal is created by intal\_create() by providing a char string of a nonnegative integer provided in decimal digits. Some intals are created out of some functionalities like intal\_add(), which creates a new intal. A new intal created have allocated dynamic memory.Responibility of destroying the intals created lies with the client by calling intal\_destroy(), which will free whetever memory allocated during the creation of intal. Client sees an intal as a “void”. It is a pointer to a structure which consists of an integer array for holding digits of the number and the length(number of digits). There is no theoretical limit to the size of the integer, but memeory limitations of the process (Operating system). If the OS allows this library is able to hold the largest prime number known , which is 23,249,425 digits long. All the functions whenever they return an intal it will stripped off the leading Zeros. Length of the integer in decimal digits is excluding the leading zeros. The null pointer is treated as not a number(NaN) in short.

**IMPLEMENTATION**

The intal is implemented using the structure called “integer” which consists of an integer array to hold the digits of the number and a length filed which indicates the number of digits in the intal.

**FUNCTIONALITIES USED**

1)void\* intal\_create (const char\* str)

This function basically takes string(array of chars with null termination) of decimal digits as parameter and converted into intal type. Input str has most significant digit at the head of the string. Void\* abstarcts out the format of intal. The returned pointer points to the intal object. Client need not know th format of the intal.The input could be a constant literal. Intal\_create() works more like atoi() which converts the intial portion of the string. That is if considers only decimal digits, and as soon as it parses a non-digit, it just processes whatever digits it has already received. If the first char is a non-digit ,it returns 0.

2) char\* intal2str(void\* intal)

This function converts intal to a string of decimal digits for mostly display purpose. Returned string has most significant non-zero digit at the head of the string.Return NaN if the intal is null.

3)void\* zeroremoval(void\* intal)

This function basically removes all the leading zeros in a number. All the functions used in this library file dealing with number without leading zeros. So every function calls this function before it process the intal.

4)void intal\_destroy(void\* intal)

This function destroys the created “object”. It mainly frees the memory allocated by the intal\_create(). It returnes nothing if the intal is null.

5)void\* intal\_increment(void\* intal)

This function increments the integer by one and returns the incremented intal. In most cases it will return the same object. But in some cases it creates a new object to accommodate the incremented value. In that case this function destroys the older intal and returns the new one.

6) void\* intal\_decrement(void\* intal)

This function decrements the integer by one and returns the decremented intal. No change if the intal is zero because it is nonnegative integer. In most of the cases It will return the same object. But in some cases it will create new object to accommodate the decremented value. In that cases this function detroys the older intal and returns the new one.

7) int intal\_compare(void\* intal1,void\* intal2)

This function takes two intals as arguments and compare them. The result depends upon the values of the intals. Returns zero when both are equal. Returns +1 when intal1 is greater ,and -1 when intal2 is greater. Returns -2 if atleast one of the input parameters are null.

8)void\* intal\_add(void\* intal1,void\* intal2)

It adds two intals and returns their sum. Here we start traversing both the array simultaneously from the end until we reach the zeroth index of either of the array. While traversing each elements of the array add elements of the both the array and carry from the previous sum. Then store the unit digit of the sum and forward carry for the next index sum. While adding zeroth index element if the carry left, then reallocate the memory using realloc function to accommodate carry also.

9)void\* intal\_diff(void\* intal1,void\* intal2)

Here difference is not subtract fucnion. Difference is always nonnegative. Difference of two nubers ‘a’ and’b’ is essentially max{a,b}-min{a,b}. there is noexplicit subtract function in this library. We are dealing with only nonnegative integers.

10) void\* intal\_multiply(void\* intal1,void\* intal2)

This function multiplies two intals and returns the product.

11) void\* intal\_divide(void\* intal1,void\* intal2)

It returns the integer part of the quotient of intal1/intal2. In this library division is implemented by using repeated subtraction. The number of time intal2 subtratcted from intal1 till it becomes less than intal2 indicates the result of division. It returns “null” if intal2 is zero. A null pointer represents NaN.

In the function the different cases considered are

* When intal2 is zero
* When intal1 is zero
* When both the intels are same
* When intal2 is greater than intal1 etc

12)void\* intal\_pow(void\* intal1,void\* intal2)

It returns intal1^intal2. It could be a really long integer for higher values of intal2. 0^n=0. This function uses divide and conquer idea. Here two cases arises

* If intal2 is even:

In this case result= ((intal1)^(intal2/2))\*2

* If intal2 is odd

Result=((intal1)^(intal2/2))\*2

**SOURCE CODE**









































