For this assignment we have two type of code: calc and format
First of all, I made it arbitrary size for calc, it will compile any size number it does not matter if it is 32-bits, but you have to preparer for a longer time for compiling the longer size integers.
The structure I used is linked list. I know that it will be easier to make it just suitable for 32-bits, but life is interesting because we have challenge. It always gave me segmentation fault, but after I work on it for a long time, I figure out that I made a lot of silly mistake. I was thinking about to give up for the unlimited size once, but I didn't. People "need" give up on some time but hope this experience will encourage me don't give up easily.

First Project

Anyway, let's talk about calc.c
I make the the structure by a node(linked list) and I just have 3 main method in my structure. 1. Change to Binary
this method is the method we need call at the beginning, it will return a binary Array pointer

2. Change From Binary

this method is a main method that will help us find the number in terms of what we want consider if the first num is 1, the number will be negative, and we need to find the first node and move to next node and make a new node suit for the type of the return type.In this case it will does not matter if the input type is octal and output type is a hex, and this will not include the binary

3. Calculation Function

this method is my main method, which will contain both plus and time minus is actually (number)+(negative number)

Now, let me explain these three function separately:
In the Change to Binary Function, I will just consider the number but will not consider the sign of the number, so all I will have is the original number without sign.
Why would I change all of them to binary? Because in that case it will be easily for octal and hex to change into binary. But not decimal. However, I was considering if I want to change them back ,the binary is the easiest way to do that and binary always have 1 or 0 it based 2 but will never exceed 1, so I used the binary number for doing this.

Then the Calculation, this main function is the biggest function in this assignment because it contains change the sign, find the same size of two binary, plus or times.
Let me explain to you separately:

Times Function (Please notice that when you try to input a times functions, you have to input like this: ./calc “*” …..)

this method is for when to number is times and right now, I will not consider the sign, after I finished the calculate I will consider the sign in the calculation method because consider 101*101 will equal to 11001 and consider that 101 is a negative number, I will still get a negative number when I square them, so I will consider the sign after this For this method, I am thinking that if I can find the number2 right most and go to left most and after I find the number I will be able to find the times */

think about the number that is not left most, what should I do? For example, if number is 10's number, the final answer will addd A zero to the end, and last calculation will or will not add the other zero to the left most(original number), so I think I will need the Same size method to compute for the original number and just simply add zero[s] to the now's number first think about the add the zero to the right now's first

Help Function

this method is a help method that will return a number(array*) that times the single number

Change The Size

this method will return a Array pointer which will be have the same size with number 1 if and only if when number1's size is bigger than number 2

Find Size

find if number 2 bigger than number 1 see how much bigger

Plus Function

consider now number 1 and number 2 have the same size also, since I already change the sign and the size and the sign, so minus will also calculate by this method and considering I will use recusal that will return the number node for the next node so it will consider that if the number is bigger than the base, so it will have the carry, and the carry will add to right now’s node(number) Then number 1 and number 2 will have the same size and this time will not consider the sign, just the number, that's it

Finally, consider the number left most node might have the number that bigger than 1 so I will consider if it is like 1-1, if it like 1-1, the carry is not zero, the number will be 0, elso I will add a node for left most to make the carry 0, because consider 1= 000......1 and -1=111...1, right now I add them together, it will be .....000.....0 and it will goes to infinity when carry is always 1; and the carry will not bigger than 2 for the next node for example 11...1+11.....1 will equal to 111...0.

So for this function, I will consider if one of their sign is + and the other one is -, in this case if the carry equal to the base, the function will goes till the end of the world, so I will just change the left most number as 0. So this will help me consider if it is such as 1-1 11+22 or 100-90. I found this three for mine project is pretty tricky. but not these will not be my problem any more.

Let me explain why:
For 1-1 for binary will be 1....000000000 and the carry always be 1. so I change the left most as zero.
For 11+22 01011+10110=100001 because it already exceed the size for number 1 and number 2, so is the first number is the sign number? not it is not, I used a trick that I will not read the sign number and I will consider after I find the number (exact number)
For 100-10 01100100+111110110=01011010 Same as number 2, the left most will exceed two, so I will add it till it not bigger than our base(which is 2), then if it still equal to the base, I will set it as zero.
Why I need consider if it is not bigger than base? Because it is possible that for a sufficiently large number, the carry for the left most will exceed the base a lot.

Change The Sign Helper

this method is a helper method that will help to change from 0 to 1 or from 1 to 0 considering this function, the change sign function will only for changing the binary's sign, so I will say that the base will always be 0. it will change the number inside, but will not change the form, also considering it will not exceed the left most number node if it exceed, in the other word when carry !=0, return 0 because it will be 0, because when the number is not zero, there must be some where that number is 1 and when you flip, the number will be 0 and it will not carry

Finally, in Calculation Function, this function is actually just change the binary back to the type what we want, 4 size of binary is hex, 3 size of binary is octal, 2's power from 0 to right most is decimal.
So I separately into two function, on is for decimal, the other one will work for the rest of the type:

FromCal

this method will return a Array pointer which is already been calculated for binary octal and hex

Dec

this method will help us get a number (Array type) in dec type, and I used a trick that I use both plus and time for calculate the power number and the summations.

2

So this is how I designed The First Project. Even though it is hard and I had take a lot of time to do the project 1, it worth it, because not only I learned a lot of things from this project but also this project have a significant influence for the Second Project.

The Second Project.

For the second project, when I saw it I felt so happy because I did it in the first project when I convert from binary to decimal.
So I "steal" it from the first project but then I found out that there are something different. The float point. I have to make it my own, the first bit of 32 bits is the sign bit 1 is negative, 0 is positive. Then there are 8 bits followed the sign bit that decide the power of two. The last 23 bit is from 1/2 1/4 1/8....
Also I need to consider that the power bits (8 bit) could have a negative number. So I need to make a double helper method that could return if it is float or an integer power number. and then I found out that for the next 23 bits it is actually ordered in this way: (1/2) (1/2)^2 (1/2)^3..... I can use the method I just made for compute the exact number (the it is 1 for binary). Then I plus all of then and finally, because it must looks in the way that (digit).(number)e(power) so I used a while loop see if it is >10 it will goes from left to right and power will be ++, other way is when it <1, the number will goes from right to left, and the power will be- -, can power will be initialized as 0. Then consider when the number is equal to 0. The loop

will never end, so write a condition sentence if it is not zero.
Finally I will be able to find out the exact number.

To readers:
Thank you for watching this paper. It only shows my logic. I know it’s a bit long and tedious. Wish you have great day!

3