```
Script started on 2019-02-15 02:04:18-0800
\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2# \007exit
README.t
[6PtGREADME.txt
pwd
/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2
\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2# pwd\033[6PtGREADME.txt
\033[0m\033[01;32mAPInt.c\033[0m
\033[01;32mAPInt.h\033[0m
\033[01;32mNoteToGrader.txt\033[0m
\033[01;32mREADME.txt\033[0m
\033[01;32mdemo.c\033[0m
\033[01;32mpa2submission.txt\033[0m
\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2# cat README.txt
C:\Users\Jeffrey\Desktop\CMPS101S18PA\CMPS101S18PA2
APInt.h -> Header file which declares the functions that will be used in my data type and h
ides the strucuter by
        assigning a pointer to the structure.

APInt.c -> Source file which implements the functions in APInt.h.

Demo.c -> Test file for APInt

NoteToGrader.txt -> A short statement about my approach to the problem.

README.txt -> List of files.\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS1
01S18PA/CMPS101S18PA2\007root@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMP
S101S18PA2# cat README.txt\033[1P\033[1P\033[1P\033[1P\033[1P\033[1P\033[1P\033[1P\033[1P\033[1P
\033[1P\033[1P\033[1P\033[1P\033[K\033[CNoteToGrader.txt
My approach:

I focus a good portion of the project learning C's syntax, primarily on how data structures
 are implemented and C,
designed my ADT similar to the First Assignment (using an Doubly Linked List). Since this w
as my first program in C, I decided to watch several tutorials C's syntax which took up a f
ew days. I then read up on how to implement Abstract Data Types in C with special emphasis
on pointers. I also figured out through a few hours of debugging that by specifically casti
ng an alias on "typedef struct Elea * APInt" actually hides from the data structure from ma
in. This is demostrated when I this:

 gcc APInt.h APInt.c demo.c
demo.c: In function \221main\222:
demo.c:34:29: error: dereferencing pointer to incomplete type \221struct Elea\222
  printf("%d\n", (*number_1) -> head -> value);

The following are tutorial's I utilized in order to build my code (Source):
 - https://www.youtube.com/watch?v=SPHcwzn6Jds&index=45&list=PLZ1QII7yudbdFfKY1eKV3x_bag04A
MPJq (series)
- https://www.edn.com/electronics-blogs/embedded-basics/4441661/5-simple-steps-to-create-an
-abstract-data-type-in-C
- https://www.youtube.com/watch?v=rk2fK2IIiiQ (series)
- https://www.youtube.com/watch?v=23mp8IUA6z4&list=LLuaXeIie13-3OwBQdcNIjKQ

The following are people how helped me debug the program:
- Alan Antonio Peral Ortiz
- Tony Dang (Senior)
- Hoa Nguyen
- Online forum



\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2# cat NoteToGrader
README.txt\033[C\033[C\033[C\033[C\033[C\033[K\033[K\033[K\033[K\033[K\033[K\033[K\033[K\033[K\033[K\033[K\033[
KA\007PInt.c\033[Kh
```

```
//Programmer: Jeffrey Wang
//CruzID: 1659820
//Data: 02.13.19
//Class: COMPS-101B (D.Bailey)

/*****************************************************************************
Programming Assignment 2: APInt Header -- Creates an interface for the APInt

CREATED: 2/11/19 12:23 P.M
Edit: 2/11/19 10:19 P.M. -- Finsished Implementation of LInked List
Edit 2/12/19 2:14 P.M -- Finsished Implementation of adder and subtractor.
Edit: 2/15/19 1:17 A.M -- Finished Documentation for all files
LinkedList Tutorial by  Jonathan Engelsma
"Learning to Program in C " - tutorial
*****************************************************************************/

#ifndef APINT_Li
#define APINT_Li

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

/*Hides Ele with APInt alias*/
typedef struct Elea * APInt;


/*Adds digit to the top of the list*/
void Add_Digit_First(APInt *list, int value);
/*Adds digits to the bottom of the list*/
void Add_Digit_Last(APInt *list, int value);


/*Intializes a zero number */
APInt* initi_APInt(void);

/*Intializes a APInt with string numerics */
APInt* initi_APInt_String(char number[]);

/*Intializes a APInt with integer type */
APInt* initi_APInt_Int(int number);

/*Free's up heap space after structure is no longer needed*/
void CleanUp(APInt *list);

/*Prints values in structure*/
void print(APInt* list);

/*Adds two APInt numbers and returns the address*/
APInt* add(APInt* adder, APInt* addend);

/*Subtracts two APInt numbers and returns the address*/
APInt* subtract(APInt* subtractor_1, APInt* subtractor_2);

/*Mulitplies two APInt numbers and returns the address*/
APInt* multiply(APInt* factor_1, APInt* factor_2);
#endif
\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2# cat APInt.h\033[K
c


//Programmer: Jeffrey Wang
//CruzID: 1659820
//Data: 02.13.19
//Class: COMPS-101B (D.Bailey)
```

```
/*******************************************************************************
Programming Assignment 2: APInt Class (Abstract Data Type)
An arbitrary precision Integer which has no fixed limit to the size of
the number. It implements a LinkedList where the nodes designates the positional
value of the digits. It contains the following methods

â\200¢a default constructor
â\200¢a constructor which uses a string, made up of optional{+,-}
followed bya string of characters from{0,1,2,3,4,5,6,7,8,9}as an input argument.
â\200¢a constructor for conversion of ints.
â\200¢a method for printing.
â\200¢methods for addition, subtraction, multiplication

CREATED: 2/11/19 12:23 P.M
Edit: 2/11/19 10:19 P.M. -- Finsished Implementation of LInked List
Edit: 2/12/19 2:14 P.M -- Finsished Implementation of adder and subtractor.
Edit: 2/121
LinkedList Tutorial by  Jonathan Engelsma
"Learning to Program in C " - tutorial
*******************************************************************************/


#include "APInt.h"
#include <string.h>

/*Defines inner structure Node type for LinkedList*/
typedef struct Nodea {

        int value;                                          //Value of Node
        struct Nodea *next;                     //Next Node
        struct Nodea *prev;                     //Previous Node

} Node;

/*Define's structure of LinkedList*/
typedef struct Elea
{
        int sign;                                           //Sign of APInt
        struct Nodea *head;                     //Head of Node
        struct Nodea *tail;                     //Tail of Node
        struct Nodea *current;          //Current Node for traversal

}Ele;

/*Adds value to the APInt as the most signficant digit*/
void Add_Digit_First(APInt * list, int data)
{
        //Intializes and allocates memory to a Node
        Node *new_digit;
        new_digit = malloc(sizeof(Node));
        new_digit -> value = data;
        new_digit -> next = NULL;
        new_digit -> prev = NULL;

        //If the LinkedList has no digits assign it as a first and have head, tail,
        //And current point to it.
        if((*list) -> head == NULL)
        {
                (*list) -> head = new_digit;
                (*list) -> tail = new_digit;
                (*list) -> current = new_digit;
        }
        //Else add new value to the head
        else
        {
                new_digit -> next = (*list) -> head;
                (*list) -> head -> prev = new_digit;
                (*list) -> head = new_digit;
```

```
        }
}

/*Adds value to the APInt as the least significant digit*/
void Add_Digit_Last(APInt * list, int data)
{
        //Intializes and allocates memory to a Node
        Node *new_digit;
        new_digit = malloc(sizeof(Node));
        new_digit -> value = data;
        new_digit -> next = NULL;
        new_digit -> prev = NULL;

        //If the LinkedList has no digits assign it as a first and have head, tail,
        //And current point to it.
        if((*list) -> tail == NULL)
        {
                (*list) -> head = new_digit;
                (*list) -> tail = new_digit;
                (*list) -> current = new_digit;
        }
        //Else add new value to the tail
        else
        {
                new_digit -> prev = (*list) -> tail;
                (*list) -> tail -> next = new_digit;
                (*list) -> tail = new_digit;
        }
}

/*This function allocates a new Ele type which is the APInt object*/
void allocate_APInt (APInt* list)
{
        APInt new_ele;
        new_ele = malloc(sizeof(Ele));
        new_ele -> sign = 1;
        new_ele -> head = NULL;
        new_ele -> tail = NULL;
        new_ele -> current = NULL;
        (*list) = new_ele;
}

/*Intializes a zero element LinkedList*/
APInt* initi_APInt(void)
{
        //Allocate memory to the pointer of pointer to APInt to store the pointer
        APInt *list =  malloc(sizeof(APInt));
        allocate_APInt(list);

        //Adds zero to the list
        Add_Digit_First(list, 0);
        return list;
}

/*Intializes a LinkedList with a string of digits.*/
/*The sign is optional*/
APInt* initi_APInt_String(char number[])
{
        /*Allocate memory to list*/
        APInt *list =  malloc(sizeof(APInt));
        allocate_APInt(list);

        //Create a pointer to string that holds to parameter
        char* str_ref;

        //Check whether or not the string has a sign.
        if((number[0] == '+') || number[0] == '-')
        {
```

```
                if(number[0] == '-')
                {
                        (*list)->sign = -1;
                }
                //Return the string of digits after the sign
                str_ref = &number[1];
        }
        else
                str_ref = number;

        //Iterate the strings and store it in list
        for (int i = 0; i < strlen(str_ref); i++)
        {
                int num =  str_ref[i] - '0';
                Add_Digit_Last(list, num);
        }
        return list;
}


/*Initializes an APInt type with an integer input*/
APInt* initi_APInt_Int(int number)
{
        /*Allocates memory to the list*/
        APInt *list =  malloc(sizeof(APInt));
        allocate_APInt(list);

        //Sets the sign of the int to a positive value
        (*list)->sign = 1;

        //If the integer is less than zero assign a negative sign
        if(number < 0)
                number *= -1;

        //Iterate through the digits in the integer
        int remainder;
        while (number != 0)
        {
                Add_Digit_First(list, (number % 10));
                number /= 10;
        }
        return list;
}

/*Frees up the allocated memory*/
void CleanUp(APInt *list)
{
        //Assigns the Nodes in List that need to be free
        Node *freeMe = (*list) -> head;
        Node *holdMe = NULL;

        //Iterate through the Nodes and frees them
        while(freeMe != NULL)
        {
                holdMe = freeMe -> next;
                free(freeMe);
                freeMe = holdMe;
        }
        // Free the allocated memory to the Ele, and APInt(pointer to Ele)
        free(*list);
        free(list);
}

/* A print function for the APInt*/
void print(APInt* list)
{
        //Assign the current Node of APInt to head for the start of a tranversal
        (*list) -> current = (*list) -> head;
```

```
        //Prints the sign
        if((*list) -> sign == 1)
                printf("%s", "+" );
        else
                printf("%s", "-");

        //Iterate through the nodes and print the value
        while((*list) -> current != NULL)
        {
                printf("%d", (*list) -> current -> value);
                (*list) -> current  = (*list) -> current -> next;
        }
        printf("\n");

}

/*Compares the value of the APInt numbers*/
bool compareTo(APInt* list1, APInt* list2)
{
        //Assign pointers to the list heads for transveral
        Node *current_1 = (*list1) -> head;
        Node *current_2 = (*list2) -> head;

        //Checks whether one APInt has more digits than the other to check value
        //Returns true if list1 has more digits than list2 (vice versa.)
        while(current_1 != NULL || current_2 != NULL)
        {
                if(current_1 == NULL && current_2 != NULL)
                        return false;
                else if(current_1 != NULL && current_2 == NULL)
                        return true;

                current_1 = current_1 -> next;
                current_2 = current_2 -> next;
        }

        //Reassigns transerval Nodes back to head if digits are the same
        current_1 = (*list1) -> head;
        current_2 = (*list2) -> head;

        //Checks each digit from MSB to LSB to check value comparision
        while(current_1 != NULL)
        {
                if(current_1 -> value > current_2 -> value)
                        return true;

                current_1 = current_1 -> next;
                current_2 = current_2 -> next;
        }
        return false;
}

/*Predefines the subtract function for add*/
APInt* subtract(APInt* subtractor_1, APInt* subtractor_2);


/*Add Function*/
APInt* add(APInt* adder, APInt* addend)
{
        if ((*adder) -> sign == (*addend) -> sign)
        {
                APInt *sum;
                sum = malloc(sizeof(APInt));
                allocate_APInt(sum);
                (*sum) -> sign = (*adder) ->sign;

                Ele *biggerAddend;
```

```
                Ele *smallerAddend;

                if(compareTo(adder, addend))
                {
                        biggerAddend = (*adder);
                        smallerAddend = (*addend);
                }
                else
                {
                        biggerAddend = (*addend);
                        smallerAddend = (*adder);
                }

                Node *current_1 = biggerAddend -> tail;
                Node *current_2 = smallerAddend -> tail;

                int carry_over = 0;
                while(current_2 != NULL)
                {
                        int total = current_1 -> value + current_2 -> value + carry_over;
                        carry_over = 0;
                        if (total >= 10)
                        {
                                carry_over = total/10;
                                total %= 10;
                        }
                        Add_Digit_First(sum, total);
                        current_1 = current_1 -> prev;
                        current_2 = current_2 -> prev;
                }

                while(current_1 != NULL)
                {
                        Add_Digit_First(sum, (current_1 -> value + carry_over));
                        carry_over = 0;
                        current_1 = current_1 -> prev;
                }

                if(carry_over != 0)
                        Add_Digit_First(sum, carry_over);

                return sum;
        }
        else
        {
                APInt *new_addend = malloc(sizeof(APInt));
                allocate_APInt(new_addend);
                 (*new_addend) -> sign = -1*((*addend)-> sign);
                 (*new_addend) -> head = (*addend) -> head;
                 (*new_addend) -> tail = (*addend) -> tail;
                 (*new_addend) -> current = NULL;
                return subtract(adder, new_addend);
        }
}

APInt* subtract(APInt* subtractor_1, APInt* subtractor_2)
{
        if((*subtractor_1) -> sign == (*subtractor_2) -> sign)
        {
                Ele minuend = {(-1 * (*subtractor_2) -> sign),
                                        (*subtractor_2) -> head,
                                        (*subtractor_2) ->tail, NULL};
                Ele     subtrahend = {(*subtractor_1) -> sign,
                                        (*subtractor_1) -> head,
                                        (*subtractor_1) ->tail, NULL};

                if(compareTo(subtractor_1, subtractor_2))
                {
```

```
                        minuend.sign = (*subtractor_1) -> sign;
                        minuend.head = (*subtractor_1) -> head;
                        minuend.tail = (*subtractor_1) -> tail;
                        subtrahend.sign = (*subtractor_2) -> sign;
                        subtrahend.head = (*subtractor_2) -> head;
                        subtrahend.tail = (*subtractor_2) ->tail;
                }

                APInt * diff = malloc(sizeof(APInt));
                allocate_APInt(diff);

                Node *current_1 = minuend.tail;
                Node *current_2 = subtrahend.tail;

                int temp = 0;
                while(current_2 != NULL)
                {
                        int difference = (current_1 -> value - temp) - current_2 -> value;
                        temp = 0;

                        if(difference < 0)
                        {
                                temp++;
                                difference += 10;
                        }

                        Add_Digit_First(diff, difference);
                        current_1 = current_1 -> prev;
                        current_2 = current_2 -> prev;
                }

                while(current_1 != NULL)
                {
                        Add_Digit_First(diff, current_1 -> value - temp);
                        temp = 0;
                        current_1 = current_1 -> prev;
                }

                while(((*diff)->head->value == 0) && (*diff)->head->next != NULL)
                {
                        (*diff)->head = (*diff)->head->next;
                        (*diff)->head->prev = NULL;
                }

                return diff;
        }
        else
        {
                APInt *new_subtractor = malloc(sizeof(APInt));
                allocate_APInt(new_subtractor);
                (*new_subtractor) -> sign =  -1*((*subtractor_2) -> sign);
                (*new_subtractor) -> head =  (*subtractor_2) -> head;
                (*new_subtractor) -> tail = (*subtractor_2) -> tail;
                (*new_subtractor) -> current = NULL;
                //Need to free new_subtractor;
                return add(subtractor_1, new_subtractor);
        }
}

APInt* multiply(APInt* factor_1, APInt* factor_2)
{
        APInt *product = malloc(sizeof(APInt));
        allocate_APInt(product);

        Node *current_1 = (*factor_1)->tail;
        Node *current_2;

        int carry_over;
```

```
        APInt *placeHolder = malloc(sizeof(APInt));
        allocate_APInt(placeHolder);

        while(current_1 != NULL)
        {
                APInt *tempPlaceHolder = malloc(sizeof(APInt));
                allocate_APInt(tempPlaceHolder);

                current_2 = (*factor_2)->tail;
                carry_over = 0;
                while(current_2 != NULL)
                {
                        int dig = (current_1 -> value * current_2 -> value + carry_over);
                        carry_over = 0;

                        if(dig >= 10)
                        {
                                Add_Digit_First(tempPlaceHolder, (dig % 10));
                                carry_over = dig/10;
                        }
                        else
                                Add_Digit_First(tempPlaceHolder, (dig));

                        current_2 = current_2 -> prev;
                }
                if(carry_over != 0)
                {
                        Add_Digit_First(tempPlaceHolder, carry_over);
                }

                Node *place_current = (*placeHolder)->head;

                while(place_current != NULL)
                {
                        Add_Digit_Last(tempPlaceHolder, 0);
                        place_current = place_current -> next;
                }


                APInt * update_product = add(product, tempPlaceHolder);
                CleanUp(product);
                product = update_product;

                Add_Digit_First(placeHolder, 0);
                CleanUp(tempPlaceHolder);
                current_1 = current_1 -> prev;
        }
        CleanUp(placeHolder);

        if((*factor_1) -> sign != (*factor_2) -> sign)
        {
                (*product) -> sign = -1;
        }
        else
        {
                (*product) -> sign = 1;
        }

        return product;
}

\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2# cat APInt.c
\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2# ls -1
\033[0m\033[01;32mAPInt.c\033[0m
```

```
\033[01;32mAPInt.h\033[0m
\033[01;32mAPInt.h.gch\033[0m
\033[01;32mNoteToGrader.txt\033[0m
\033[01;32mREADME.txt\033[0m
\033[01;32mdemo.c\033[0m
\033[01;32mpa2submission.txt\033[0m
\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2# gc\033[K033[K033
[Kcat demo.c

//Programmer: Jeffrey Wang
//CruzID: 1659820
//Data: 02.13.19
//Class: COMPS-101B (D.Bailey)

/*********************************************************************************
Programming Assignment 2: Demo -- Insures that the ADT's are working
correctly is working correctly

CREATED: 2/11/19 12:23 P.M
Edit: 2/11/19 10:19 P.M. -- Finsished Implementation of LInked List
Edit 2/12/19 2:14 P.M -- Finsished Implementation of adder and subtractor.
Edit: 2/15/19 1:17 A.M -- Finished Documentation for all files
LinkedList Tutorial by  Jonathan Engelsma
"Learning to Program in C " - tutorial
*********************************************************************************/

/*
All Calculations are double checked with programming assignment 1*/

#include <stdio.h>
#include "APInt.h"
int main(void)
{
        APInt* number_1 = initi_APInt();
        APInt* number_2 = initi_APInt_String("-271828182845904523536028747135
2");
        APInt* number_3 = initi_APInt_String("141421356237309504880168872420969807856967187
5");
        APInt* number_4 = initi_APInt_Int(-123456);


        printf("Representation of No-Arg Constructor for APInt (number_1): \n ");
        print(number_1);
        printf("\n");

        printf("Representation of String Conversion (with sign) to APInt (number_2): \n ");
        print(number_2);
        printf("\n");

        printf("Representation of String Conversion (without sign) to APInt (number_3):  \n
 ");
        print(number_3);
        printf("\n");

        printf("Representation of Integer Conversion (number_4): \n ");
    print(number_4);
        printf("\n");

        printf("num4 + num1(0):\n");
        APInt* sum0 = add(number_4, number_1);
        print(sum0);
        printf("\n");
        CleanUp(sum0);

        printf("num2 + num3:\n");
        APInt* sum1 = add(number_2, number_3);
        print(sum1);
        printf("\n");
```

```
        CleanUp(sum1);

        printf("num2 + num4:\n");
        APInt* sum2 = add(number_2, number_3);
        print(sum2);
        printf("\n");
        CleanUp(sum2);


        printf("num2 - num3:\n");
        APInt* difference1 = subtract(number_2, number_3);
        print(difference1);
        printf("\n");
        CleanUp(difference1);

        printf("num4 - num2:\n");
        APInt* difference2 = subtract(number_4, number_2);
        print(difference2);
        printf("\n");
        CleanUp(difference2);


        printf("num2 * num3:\n");
        APInt* product = multiply(number_2, number_3);
        print(product);
        printf("\n");
        CleanUp(product);

        CleanUp(number_1);
        CleanUp(number_2);
        CleanUp(number_3);
        CleanUp(number_4);
}
\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2# gcc AP\007Int.c
Aa\033[K\033[Kdemo.c
\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2# ./a.out
Representation of No-Arg Constructor for APInt (number_1):
 +0

Representation of String Conversion (with sign) to APInt (number_2):
 -271828182845904523536028747135 2

Representation of String Conversion (without sign) to APInt (number_3):
 +1414213562373095048801688724209698078569671875

Representation of Integer Conversion (number_4):
 +123456

num4 + num1(0):
+123456

num2 + num3:
+1414213562373092330519860265164462718282200523

num2 + num4:
+1414213562373092330519860265164462718282200523

num2 - num3:
-1414213562373097767083517183254933438857143227

num4 - num2:
+271828182845904523536028759 4808

num2 * num3:
-384423102815911682486367163742533996467485780164417401505550572580010262 5000
```

```
\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA2# exit
exit

Script done on 2019-02-15 02:06:25-0800
```