Department of Computer Science

CO24497 - Programming Practices

MINI PROJECT

Author:

Shashank Pardeshi

Enrolment Number : 0801CS211077

Date : 21-11-2022

Contents

Part I

INFORMATION ABOUT PROJECT

0.1 • Objective

This project is created to do the arithmetic operations(like multiplication , addition , conversion ,etc) on numbers of any base.

0.2 • Statistics

- Start date : 18-11-22
- Finish date : 21-11-22
- total number of functions in the program : 13
- total lines of code in the program : 360

Part II

DESCRIPTION OF THE FUNCTIONS

• Function-1

Function name : user_guide

This function is used to print the user manual. It will tell the user which number should he/she enter to perform which operations.

• Function-2

This function is used to get the numerical value associated with the alphabets of number with the base greater than 10. This function takes a character as argument and check whether it is a number or a alphabet. If it is a number the function will return that numbers as it is but if the character is a alphabet the function will return the value associated with that alphabet.

• Function-3

Function name : convert_to_decimal

This function will be used to convert a number in any base to a decimal number.

This function takes a string of characters and a integer value as arguments. The string of character that the function is receiving is the number which has to converted into its decimal equivalent form. The other argument is used to tell the function the base of the number which has to converted to decimal.

• Function-4

Function name : return_character

This function is used to get the character which is associated with a numerical value in the number system. This function takes a integer value as argument and returns the character associated with that numerical value.

- **Function-5**

Function name : reverse_string

This function is a helper function which is used to reverse any string. This function takes a string and returns the reverse of that string.

- **Function-6**

Function name : convert_from_decimal

This function is used to convert a decimal number to a number of any base. This function takes three variables as arguments, one string which is used to store the the final result of the function and two integers, one to receive the number to be converted and other to receive the base to which the given number has to be converted.

This function takes help of return$_{c}haracterfunctiontogetthecha$

- **Function-7**

Function name : multiplier

This function is used to do the multiplication of two numbers of any base. It takes a total of three inputs. Two strings of character and a integer.

It first converts the two numbers given to it as string to the decimal numbers (using function number 3) and then do the multiplication of those two decimal numbers and then convert the result of the multiplication to the actual base of the numbers (using function number 6).

- ## Function-8

Function name : divide

This function do the division of two numbers of any base. It also works the same way as the multiplier function works. Takes three inputs, two number to do the division and a integer to know the base of those numbers. This function also converts the two numbers to decimal and then do the division and convert them back their original base.

- ## Function-9

Function name : addition

This function is used to do the addition of two numbers. It takes two numbers and then converts them to decimal. Then it do their addition and then convert them back to their original base. It calls the convert_to_decimal function twice and convert_from_decimal once to complete its task.

- ## Function-10

Function name : subtraction

This function is used to do the subtraction of two numbers of some base. It is similar to the function "addition", the only difference is this function do the subtraction of numbers instead of addition.

- ## Function-11

Function name : complement

This function prints the complement of any number. It takes a string of character and a integer value as arguments. It subtract each digit of the number given to this function and store the difference to the string and then uses the function "return_character" to print the complement of the given number. It uses

a for loop to print the elements of the string without any spaces between them so that they appear as a single number.

- ## Function-12

Function name : value_of_power

This function is used to evaluate the value of any number of any base raised to some power. It first converts the number given to it as argument to decimal number and then raise that decimal number to given power and then convert the result to the base of the given number.

- ## Function-13

Function name : modulo_finder

This function is used to find the remainder that is left when we divide a number to another number.

Part III

CODE OF THE PROJECT

```
#include <stdio.h>
#include <string.h>
#include <math.h>

/*
 Funtion 1
 This function will print the
 user guide which will help the user
 */
void user_guide(){
 printf( "\n" ) ;
printf( "Enter 1 : to convert a number in one base to any other base \n" )
printf( "Enter 2 : to multiply two numbers of any base \n" ) ;
printf( "Enter 3 : to divide two numbers of any base \n" ) ;
printf( "Enter 4 : to add two numbers of any base \n" ) ;
printf( "Enter 5 : to subtract two numbers of any base \n" ) ;
printf( "Enter 6 : to find the complement of the number in any base \n" ) ;
printf( "Enter 7 : to find the value of any number in any base raised to so
printf( "Enter 8 : to find the modulo of any division \n" ) ;
}

/*
Funtion 2
This function will return the numerical
value of the character received
by the function
*/
int value_of_character (char c){
    if ( c >= '0' && c <= '9' ){
        return (int)c - '0' ;
    } else if ( c >= 'a' && c <= 'z' ){
        return (int)c - 'a' + 10 ;
    } else{
        return (int)c - 'A' + 10 ;
    }
}

/*
Funtion 3
This function will convert a number
from any base to decimal
*/
int convert_to_decimal ( char *str , int base ){
    int len = strlen( str ) ;
    int power = 1 ;
    int to_decimal = 0 ;
    for (int i = len - 1 ; i >= 0 ; i--){

        if (value_of_character(str[i]) >= base){
           printf( "Invalid Number" ) ;
           return -1 ;
        }

        to_decimal += value_of_character(str[i]) * power ;
        power = power * base ;
    }

    return to_decimal ;
}

/*
Function 4
This function will return the
the character associated with the
number received by it
*/
char return_character(int num){
    if ( num >= 0 && num <= 9 ){
        return (char)( num + '0' ) ;
    } else if ( num >= 'a' && num <= 'z' ){
        return (char)( num - 10 + 'a' ) ;
    } else{
        return (char)( num - 10 + 'A' ) ;
    }
}

/*
Function 5
This function will reverse an array
*/
void reverse_string( char *str ){

    int len = strlen( str ) ;

    for ( int i = 0; i < len/2; i++ ){
        char temp = str[i] ;
        str[i] = str[len-i-1] ;
        str[len-i-1] = temp ;
    }
}

/*
function 6
```

```
This function will convert any number
from decimal to any base
*/
char* convert_from_decimal( char *res , int base , int inputNum ){
    int index = 0 ;  // Initialize index of result

    while ( inputNum > 0 ){
        res[index] = return_character( inputNum % base ) ;
        inputNum /= base ;
        index++ ;
    }

    res[index] = '\0' ;

    // Reverse the result
    reverse_string(res) ;

    return res ;
}

/*
Function 7
This function will multiply two numbers
of any base
*/
void multiplier( char* x , char* y , int base ){

    int num_1 = convert_to_decimal( x , base ) ;
    int num_2 = convert_to_decimal( y , base ) ;

    int result = num_1 * num_2 ;

    char new_string[100] ;

    convert_from_decimal( new_string , base , result ) ;
    printf( "The result is : %s \n" , new_string ) ;

}

/*
Function 8
This function will divide two numbers
of any base
*/
void divide( char* x , char* y , int base ){

    int num_1 = convert_to_decimal( x , base ) ;
    int num_2 = convert_to_decimal( y , base ) ;
    int result = num_1 / num_2 ;

    char new_string[100] ;
    convert_from_decimal( new_string , base , result ) ;
    printf( "The result is : %s \n" , new_string ) ;

}

/*
Function 9
This function will add two numbers
of any base
*/
void addition( char* x , char* y , int base ){

    int num_1 = convert_to_decimal( x , base ) ;
    int num_2 = convert_to_decimal( y , base ) ;

    int result = num_1 + num_2 ;

    char new_string[100] ;

    convert_from_decimal( new_string , base , result ) ;
    printf( "The result is : %s \n" , new_string ) ;

}

/*
Function 10
This function will subtract two numbers
of any base
*/
void subtraction( char* x , char* y , int base ){

    int num_1 = convert_to_decimal( x , base ) ;
    int num_2 = convert_to_decimal( y , base ) ;

    int result = num_1 - num_2 ;

    char new_string[100] ;

    convert_from_decimal( new_string , base , result ) ;
    printf( "The result is : %s \n" , new_string ) ;

}

/*
```

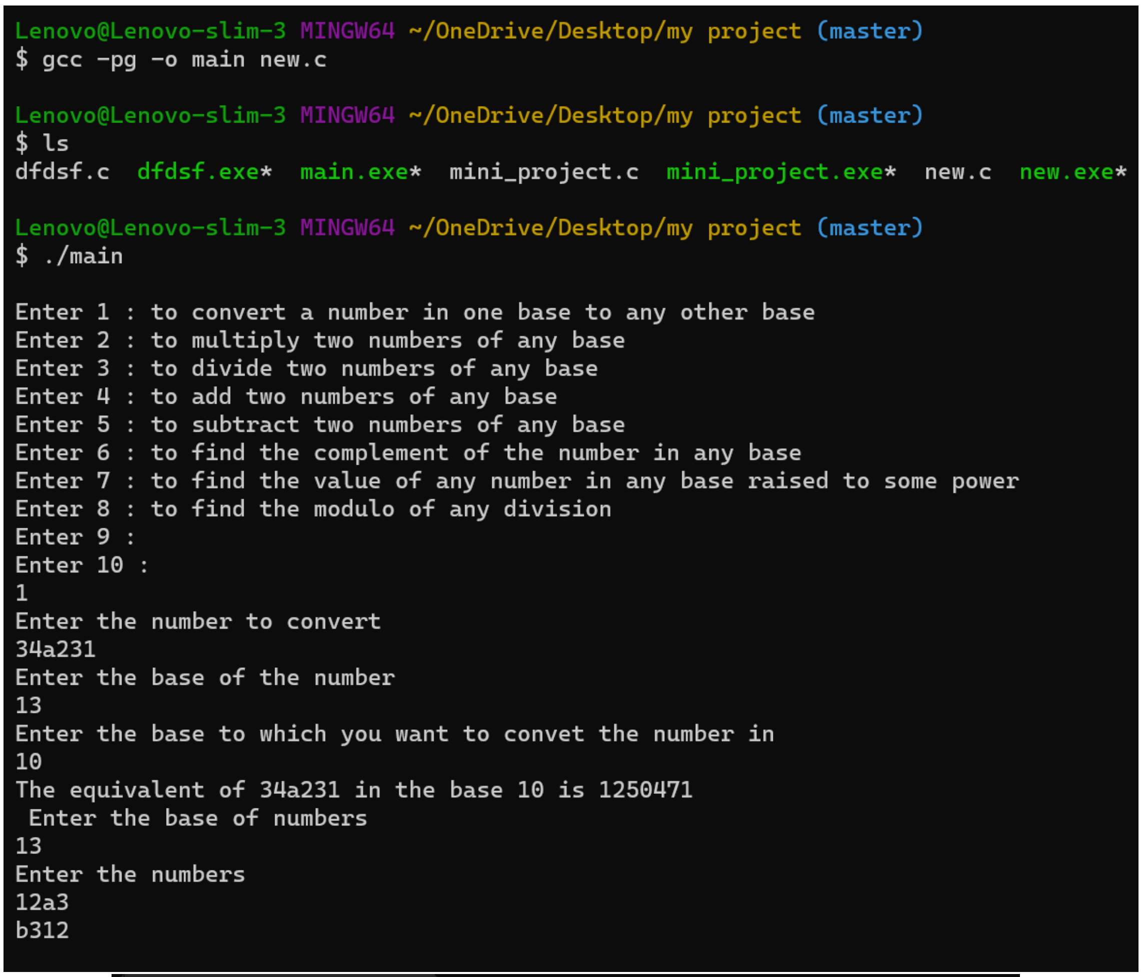
```
function 11
This function will return the complement
of any number in any base
*/
void complement( char *str , int base ){
    int len = strlen(str) ;
    int result_string[len] ;
    for ( int i = len - 1 ; i >= 0 ; i-- ){
        result_string[i] = ( base - 1 ) - value_of_character(str[i]) ;
        result_string[i] = return_character( result_string[i] ) ;
    }

    for ( int i = 0 ; i < len ; i++ ){
        printf( "%c" , result_string[i] ) ;
    }
    printf( "\n" ) ;
}

/*
Function 12
This function will evaluate the value
of any number raised to some power
*/
void value_of_power( char *str , int power , int base ){

    int decimal = convert_to_decimal( str , base ) ;
    decimal = pow( decimal , power ) ;

    char new_string[100] ;

    convert_from_decimal( new_string , base , decimal ) ;

    printf( "The result is : %s \n" , new_string ) ;
}

/*
Function 13
This function will find the modulo of
any division
*/
void modulo_finder( char *str_1 , char *str_2 , int base ){
    int num_1 = convert_to_decimal( str_1 , base ) ;
    int num_2 = convert_to_decimal( str_2 , base ) ;

    int result = num_1 % num_2 ;

    char new_string[100] ;
    convert_from_decimal( new_string , base , result ) ;
    printf( ”The result is : %s \n” , new_string ) ;
}



int main(){

    char string_of_digits[100] ;
    char string_of_result[100] ;
    char string_1[100] ;
    char string_2[100] ;
    int initial_base ;
    int final_base ;
    int selector ;
    int num_1 ;

    user_guide() ;

    scanf(”%d” , &selector) ;

    switch(selector){
        case 1:

        printf(”Enter the number to convert \n”);
        scanf(”%s”, string_of_digits);
        printf(”Enter the base of the number \n”);
        scanf(”%d”, &initial_base);
        printf(”Enter the base to which you want to convet the number in \
        scanf(”%d”, &final_base);

        int decimal_value = convert_to_decimal(string_of_digits , initial_b

        convert_from_decimal( string_of_result , final_base , decimal_value

        printf(”The equivalent of %s in the base %d is %s ” , string_of_dig
        break ;

        case 2:

            printf(”Enter the base of numbers \n”) ;
            scanf( ”%d” , &initial_base ) ;
            printf(”Enter the numbers \n”) ;
            scanf( ”%s%s” , string_1 , string_2 ) ;

            multiplier( string_1 , string_2 , initial_base ) ;
            break ;
```

```
case 3:

    printf( "Enter the base of numbers \n ") ;
    scanf("%d" , &initial_base ) ;
    printf("Enter the divident \n" ) ;
    scanf("%s" , string_1 ) ;
    printf("Enter the divisor \n" ) ;
    scanf("%s" , string_2 ) ;

    divide( string_1 , string_2 , initial_base ) ;
    break ;

case 4:

    printf( "Enter the base of numbers \n " ) ;
    scanf("%d" , &initial_base ) ;
    printf( "Enter the numbers \n " ) ;
    scanf("%s%s" , string_1 , string_2 ) ;

    addition( string_1 , string_2 , initial_base ) ;
    break ;

case 5:

    printf( "Enter the base of numbers \n " ) ;
    scanf("%d" , &initial_base ) ;
    printf( "Enter the numbers \n " ) ;
    scanf("%s%s" , string_1 , string_2 ) ;

    subtraction( string_1 , string_2 , initial_base ) ;
    break ;

case 6:

    printf( "Enter the base of the number \n " ) ;
    scanf( "%d" , &initial_base ) ;
    printf( "Enter the number \n " ) ;
    scanf( "%s" , string_1 ) ;

    complement( string_1 , initial_base ) ;
    break ;

case 7:

    printf( "Enter the base of the number \n " ) ;
    scanf("%d" , &initial_base ) ;
    printf( "Enter the number \n " ) ;
    scanf( "%s" , string_1 ) ;
```

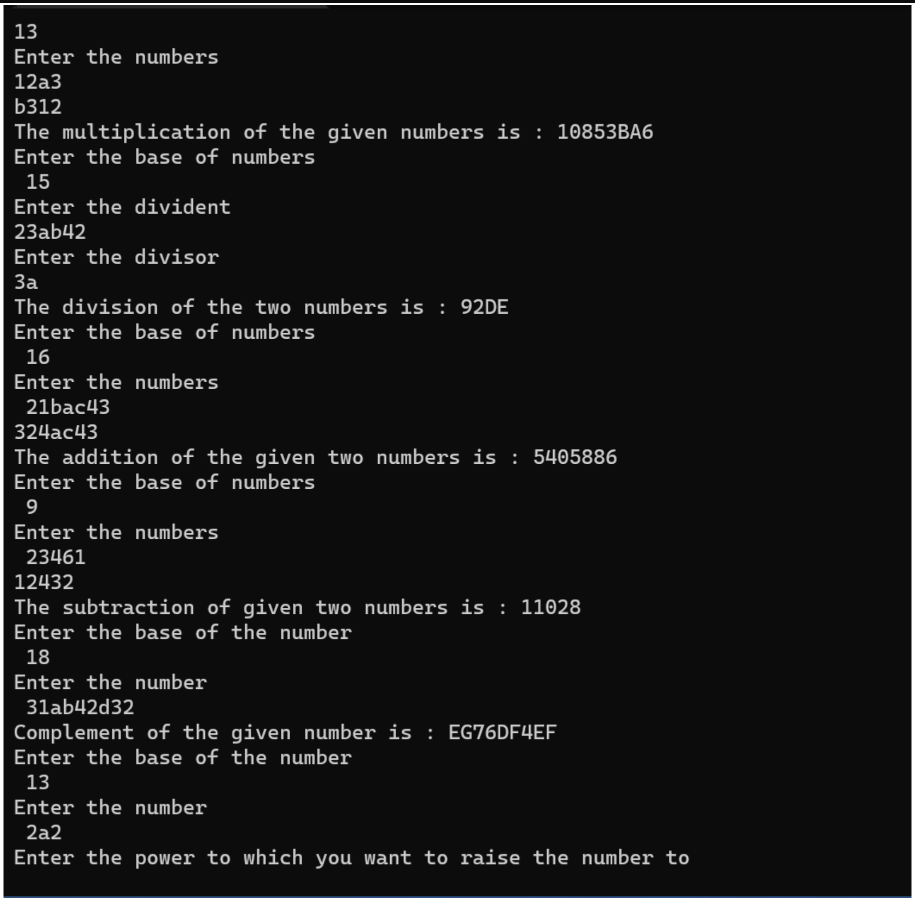
```
            printf( "Enter the power to which you want to raise the number
            scanf("%d" , &num_1 ) ;

            value_of_power( string_1 , num_1 , initial_base ) ;
            break ;

        case 8:

            printf( "Enter the base of numbers \n " ) ;
            scanf("%d" , &initial_base ) ;
            printf("Enter the divident \n " ) ;
            scanf("%s" , string_1 ) ;
            printf( "Enter the divisor \n " ) ;
            scanf("%s" , string_2 ) ;

            modulo_finder( string_1 , string_2 , initial_base ) ;
            break ;

        default :

            printf("You entered incorrect number \n " ) ;
    }

    return 0 ;
}
```

Part IV

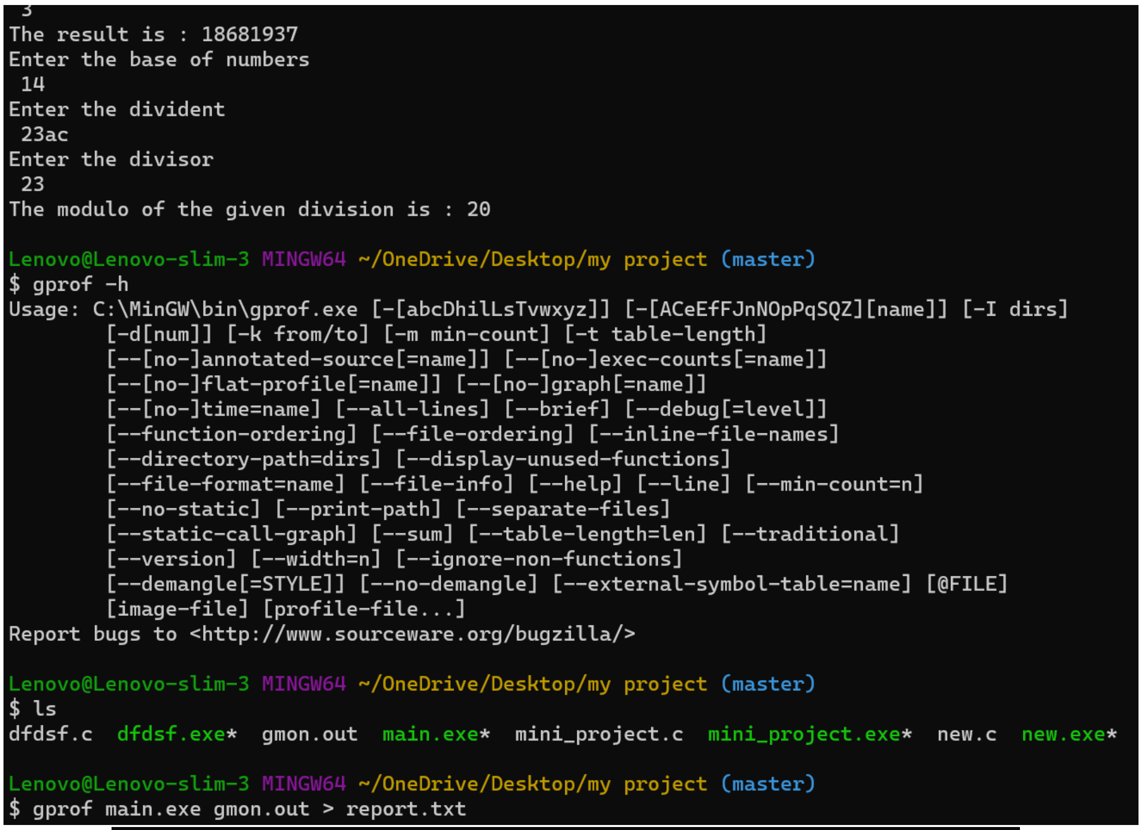
PROFILING OF THE PROGRAM



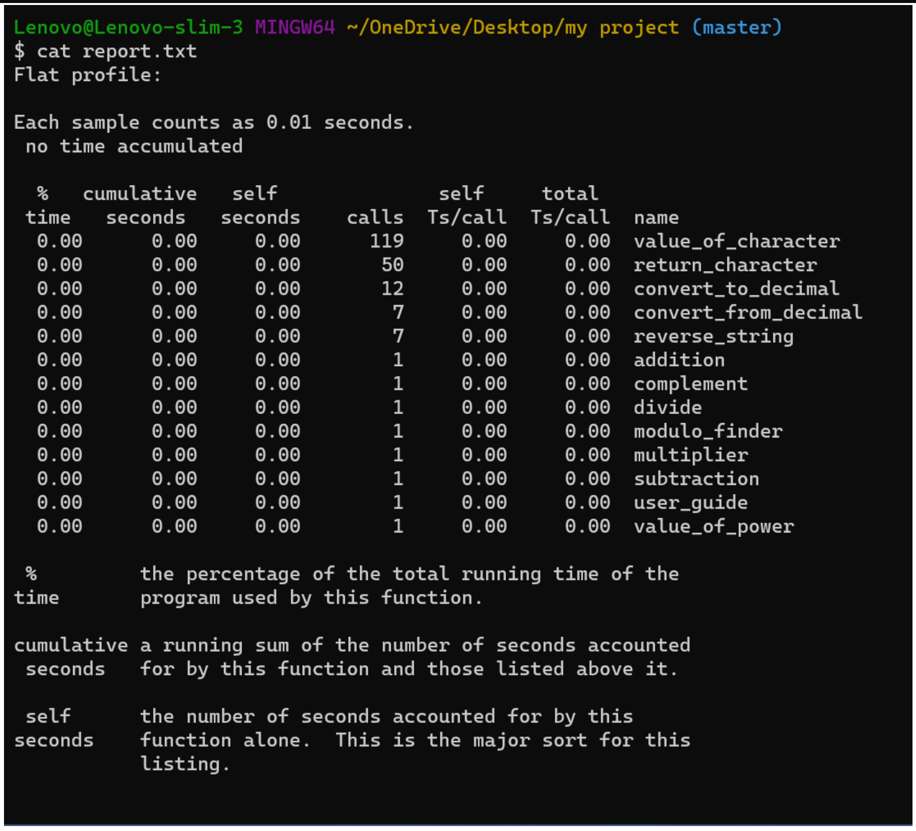
```
Lenovo@Lenovo-slim-3 MINGW64 ~/OneDrive/Desktop/my project (master)
$ gcc -pg -o main new.c

Lenovo@Lenovo-slim-3 MINGW64 ~/OneDrive/Desktop/my project (master)
$ ls
dfdsf.c  dfdsf.exe*  main.exe*  mini_project.c  mini_project.exe*  new.c  new.exe*

Lenovo@Lenovo-slim-3 MINGW64 ~/OneDrive/Desktop/my project (master)
$ ./main

Enter 1 : to convert a number in one base to any other base
Enter 2 : to multiply two numbers of any base
Enter 3 : to divide two numbers of any base
Enter 4 : to add two numbers of any base
Enter 5 : to subtract two numbers of any base
Enter 6 : to find the complement of the number in any base
Enter 7 : to find the value of any number in any base raised to some power
Enter 8 : to find the modulo of any division
Enter 9 :
Enter 10 :
1
Enter the number to convert
34a231
Enter the base of the number
13
Enter the base to which you want to convet the number in
10
The equivalent of 34a231 in the base 10 is 1250471
 Enter the base of numbers
13
Enter the numbers
12a3
b312
```



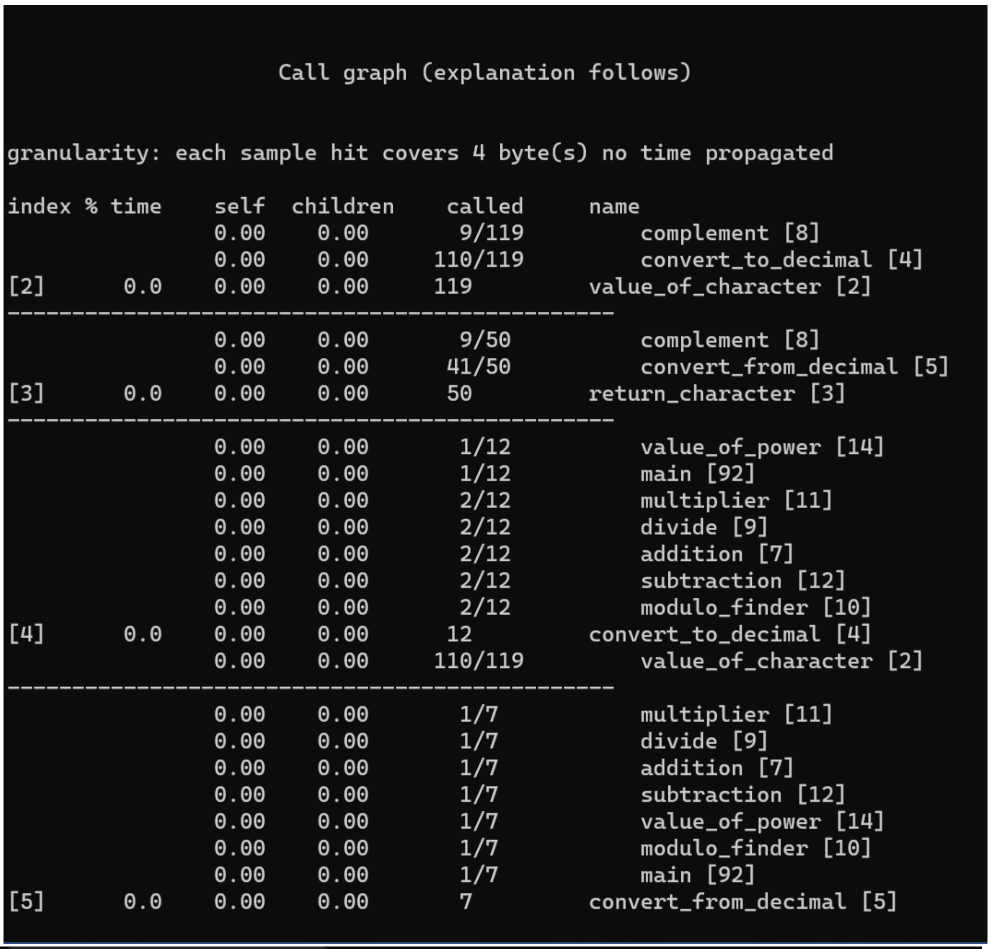
```
13
Enter the numbers
12a3
b312
The multiplication of the given numbers is : 10853BA6
Enter the base of numbers
 15
Enter the divident
23ab42
Enter the divisor
3a
The division of the two numbers is : 92DE
Enter the base of numbers
 16
Enter the numbers
 21bac43
324ac43
The addition of the given two numbers is : 5405886
Enter the base of numbers
 9
Enter the numbers
 23461
12432
The subtraction of given two numbers is : 11028
Enter the base of the number
 18
Enter the number
 31ab42d32
Complement of the given number is : EG76DF4EF
Enter the base of the number
 13
Enter the number
 2a2
Enter the power to which you want to raise the number to
```



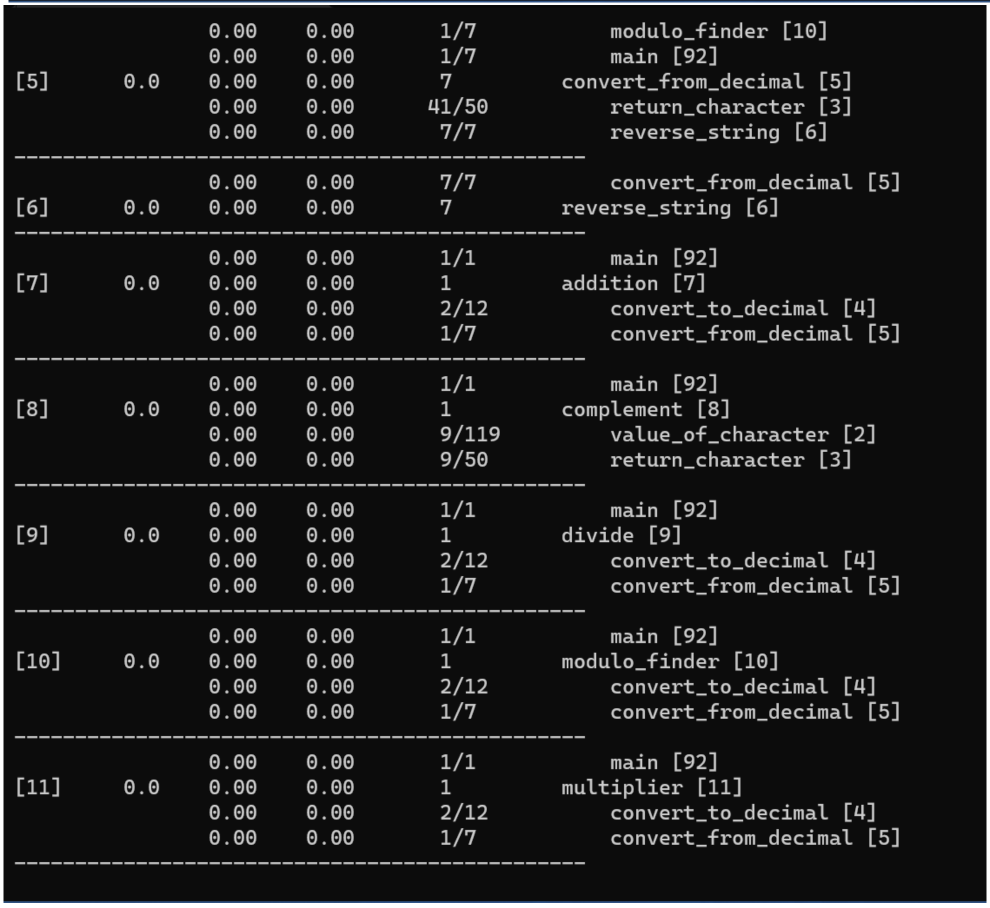
```
 3
The result is : 18681937
Enter the base of numbers
 14
Enter the divident
 23ac
Enter the divisor
 23
The modulo of the given division is : 20

Lenovo@Lenovo-slim-3 MINGW64 ~/OneDrive/Desktop/my project (master)
$ gprof -h
Usage: C:\MinGW\bin\gprof.exe [-[abcDhilLsTvwxyz]] [-[ACeEfFJnNOpPqSQZ][name]] [-I dirs]
        [-d[num]] [-k from/to] [-m min-count] [-t table-length]
        [--[no-]annotated-source[=name]] [--[no-]exec-counts[=name]]
        [--[no-]flat-profile[=name]] [--[no-]graph[=name]]
        [--[no-]time=name] [--all-lines] [--brief] [--debug[=level]]
        [--function-ordering] [--file-ordering] [--inline-file-names]
        [--directory-path=dirs] [--display-unused-functions]
        [--file-format=name] [--file-info] [--help] [--line] [--min-count=n]
        [--no-static] [--print-path] [--separate-files]
        [--static-call-graph] [--sum] [--table-length=len] [--traditional]
        [--version] [--width=n] [--ignore-non-functions]
        [--demangle[=STYLE]] [--no-demangle] [--external-symbol-table=name] [@FILE]
        [image-file] [profile-file...]
Report bugs to <http://www.sourceware.org/bugzilla/>

Lenovo@Lenovo-slim-3 MINGW64 ~/OneDrive/Desktop/my project (master)
$ ls
dfdsf.c  dfdsf.exe*  gmon.out  main.exe*  mini_project.c  mini_project.exe*  new.c  new.exe*

Lenovo@Lenovo-slim-3 MINGW64 ~/OneDrive/Desktop/my project (master)
$ gprof main.exe gmon.out > report.txt
```



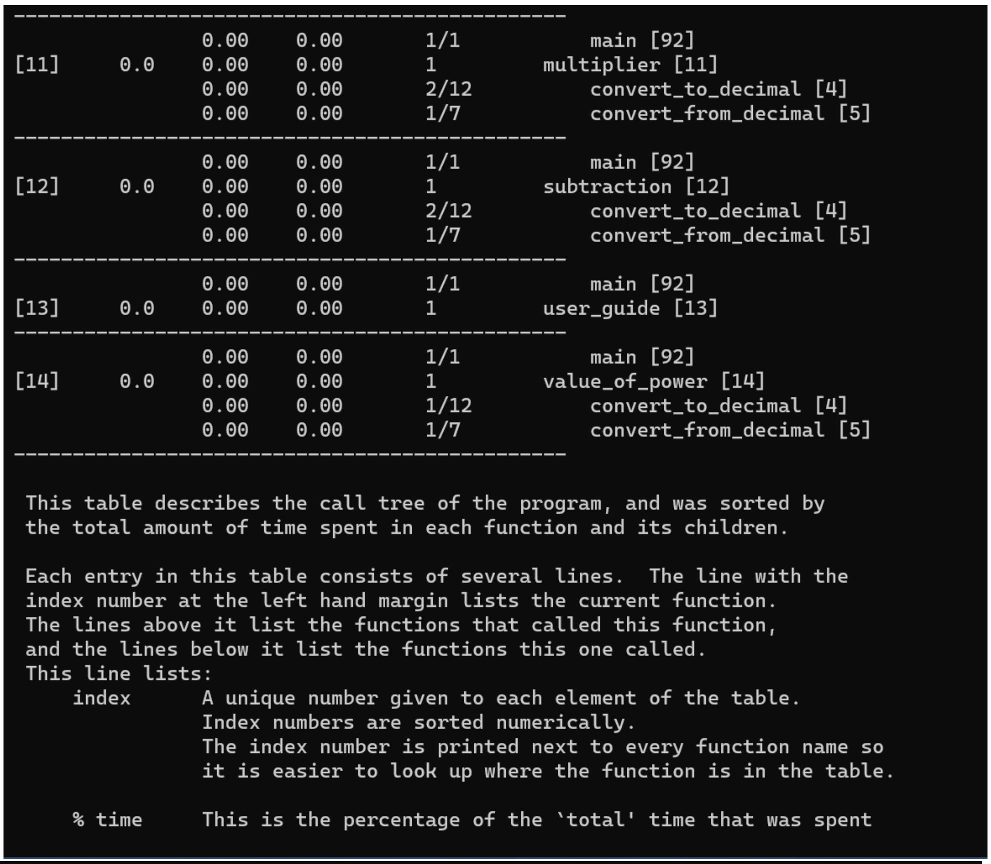
```
Lenovo@Lenovo-slim-3 MINGW64 ~/OneDrive/Desktop/my project (master)
$ cat report.txt
Flat profile:

Each sample counts as 0.01 seconds.
 no time accumulated

  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
  0.00      0.00     0.00      119     0.00     0.00  value_of_character
  0.00      0.00     0.00       50     0.00     0.00  return_character
  0.00      0.00     0.00       12     0.00     0.00  convert_to_decimal
  0.00      0.00     0.00        7     0.00     0.00  convert_from_decimal
  0.00      0.00     0.00        7     0.00     0.00  reverse_string
  0.00      0.00     0.00        1     0.00     0.00  addition
  0.00      0.00     0.00        1     0.00     0.00  complement
  0.00      0.00     0.00        1     0.00     0.00  divide
  0.00      0.00     0.00        1     0.00     0.00  modulo_finder
  0.00      0.00     0.00        1     0.00     0.00  multiplier
  0.00      0.00     0.00        1     0.00     0.00  subtraction
  0.00      0.00     0.00        1     0.00     0.00  user_guide
  0.00      0.00     0.00        1     0.00     0.00  value_of_power

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.
```



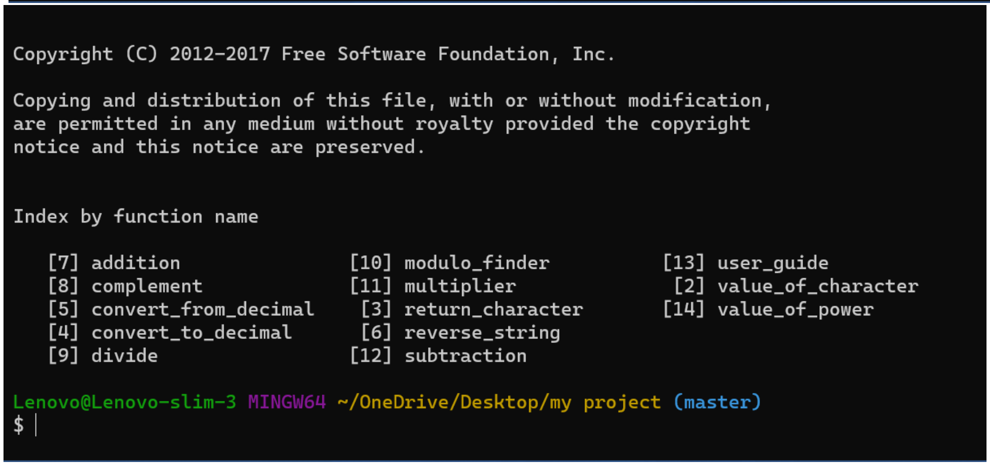
```
                     Call graph (explanation follows)


granularity: each sample hit covers 4 byte(s) no time propagated

index % time    self  children    called     name
                0.00    0.00       9/119         complement [8]
                0.00    0.00     110/119         convert_to_decimal [4]
[2]      0.0    0.00    0.00     119         value_of_character [2]
-----------------------------------------------
                0.00    0.00       9/50          complement [8]
                0.00    0.00      41/50          convert_from_decimal [5]
[3]      0.0    0.00    0.00      50         return_character [3]
-----------------------------------------------
                0.00    0.00       1/12          value_of_power [14]
                0.00    0.00       1/12          main [92]
                0.00    0.00       2/12          multiplier [11]
                0.00    0.00       2/12          divide [9]
                0.00    0.00       2/12          addition [7]
                0.00    0.00       2/12          subtraction [12]
                0.00    0.00       2/12          modulo_finder [10]
[4]      0.0    0.00    0.00      12         convert_to_decimal [4]
                0.00    0.00     110/119         value_of_character [2]
-----------------------------------------------
                0.00    0.00       1/7           multiplier [11]
                0.00    0.00       1/7           divide [9]
                0.00    0.00       1/7           addition [7]
                0.00    0.00       1/7           subtraction [12]
                0.00    0.00       1/7           value_of_power [14]
                0.00    0.00       1/7           modulo_finder [10]
                0.00    0.00       1/7           main [92]
[5]      0.0    0.00    0.00       7         convert_from_decimal [5]
```



```
                0.00    0.00       1/7           modulo_finder [10]
                0.00    0.00       1/7           main [92]
[5]      0.0    0.00    0.00       7         convert_from_decimal [5]
                0.00    0.00      41/50          return_character [3]
                0.00    0.00       7/7           reverse_string [6]
-----------------------------------------------
                0.00    0.00       7/7           convert_from_decimal [5]
[6]      0.0    0.00    0.00       7         reverse_string [6]
-----------------------------------------------
                0.00    0.00       1/1           main [92]
[7]      0.0    0.00    0.00       1         addition [7]
                0.00    0.00       2/12          convert_to_decimal [4]
                0.00    0.00       1/7           convert_from_decimal [5]
-----------------------------------------------
                0.00    0.00       1/1           main [92]
[8]      0.0    0.00    0.00       1         complement [8]
                0.00    0.00       9/119         value_of_character [2]
                0.00    0.00       9/50          return_character [3]
-----------------------------------------------
                0.00    0.00       1/1           main [92]
[9]      0.0    0.00    0.00       1         divide [9]
                0.00    0.00       2/12          convert_to_decimal [4]
                0.00    0.00       1/7           convert_from_decimal [5]
-----------------------------------------------
                0.00    0.00       1/1           main [92]
[10]     0.0    0.00    0.00       1         modulo_finder [10]
                0.00    0.00       2/12          convert_to_decimal [4]
                0.00    0.00       1/7           convert_from_decimal [5]
-----------------------------------------------
                0.00    0.00       1/1           main [92]
[11]     0.0    0.00    0.00       1         multiplier [11]
                0.00    0.00       2/12          convert_to_decimal [4]
                0.00    0.00       1/7           convert_from_decimal [5]
-----------------------------------------------
```



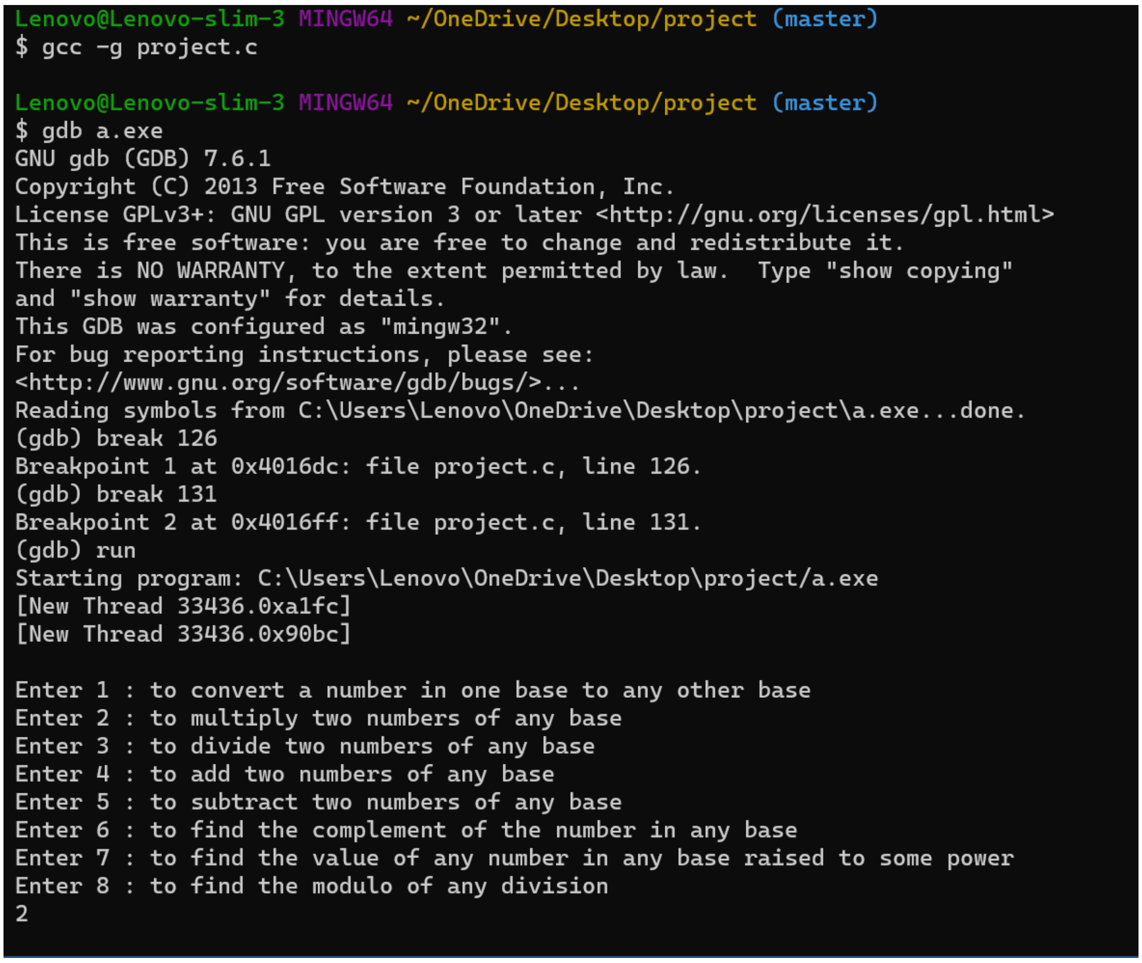
```
-----------------------------------------------
                0.00    0.00       1/1           main [92]
[11]     0.0    0.00    0.00       1         multiplier [11]
                0.00    0.00       2/12          convert_to_decimal [4]
                0.00    0.00       1/7           convert_from_decimal [5]
-----------------------------------------------
                0.00    0.00       1/1           main [92]
[12]     0.0    0.00    0.00       1         subtraction [12]
                0.00    0.00       2/12          convert_to_decimal [4]
                0.00    0.00       1/7           convert_from_decimal [5]
-----------------------------------------------
                0.00    0.00       1/1           main [92]
[13]     0.0    0.00    0.00       1         user_guide [13]
-----------------------------------------------
                0.00    0.00       1/1           main [92]
[14]     0.0    0.00    0.00       1         value_of_power [14]
                0.00    0.00       1/12          convert_to_decimal [4]
                0.00    0.00       1/7           convert_from_decimal [5]
-----------------------------------------------

 This table describes the call tree of the program, and was sorted by
 the total amount of time spent in each function and its children.

 Each entry in this table consists of several lines.  The line with the
 index number at the left hand margin lists the current function.
 The lines above it list the functions that called this function,
 and the lines below it list the functions this one called.
 This line lists:
     index      A unique number given to each element of the table.
                Index numbers are sorted numerically.
                The index number is printed next to every function name so
                it is easier to look up where the function is in the table.

     % time     This is the percentage of the `total' time that was spent
```



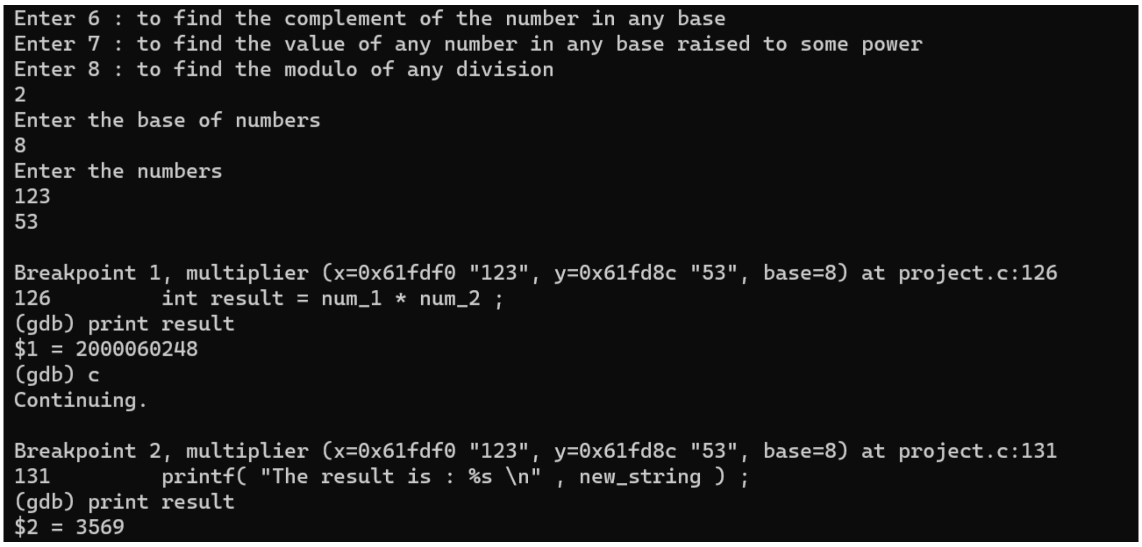
```
Copyright (C) 2012-2017 Free Software Foundation, Inc.

Copying and distribution of this file, with or without modification,
are permitted in any medium without royalty provided the copyright
notice and this notice are preserved.


Index by function name

   [7] addition                [10] modulo_finder           [13] user_guide
   [8] complement              [11] multiplier               [2] value_of_character
   [5] convert_from_decimal     [3] return_character        [14] value_of_power
   [4] convert_to_decimal       [6] reverse_string
   [9] divide                  [12] subtraction

Lenovo@Lenovo-slim-3 MINGW64 ~/OneDrive/Desktop/my project (master)
$ |
```

Part V

DEBUGGING OF THE PROGRAM



```
Lenovo@Lenovo-slim-3 MINGW64 ~/OneDrive/Desktop/project (master)
$ gcc -g project.c

Lenovo@Lenovo-slim-3 MINGW64 ~/OneDrive/Desktop/project (master)
$ gdb a.exe
GNU gdb (GDB) 7.6.1
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "mingw32".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from C:\Users\Lenovo\OneDrive\Desktop\project\a.exe...done.
(gdb) break 126
Breakpoint 1 at 0x4016dc: file project.c, line 126.
(gdb) break 131
Breakpoint 2 at 0x4016ff: file project.c, line 131.
(gdb) run
Starting program: C:\Users\Lenovo\OneDrive\Desktop\project/a.exe
[New Thread 33436.0xa1fc]
[New Thread 33436.0x90bc]

Enter 1 : to convert a number in one base to any other base
Enter 2 : to multiply two numbers of any base
Enter 3 : to divide two numbers of any base
Enter 4 : to add two numbers of any base
Enter 5 : to subtract two numbers of any base
Enter 6 : to find the complement of the number in any base
Enter 7 : to find the value of any number in any base raised to some power
Enter 8 : to find the modulo of any division
2
```



```
Enter 6 : to find the complement of the number in any base
Enter 7 : to find the value of any number in any base raised to some power
Enter 8 : to find the modulo of any division
2
Enter the base of numbers
8
Enter the numbers
123
53

Breakpoint 1, multiplier (x=0x61fdf0 "123", y=0x61fd8c "53", base=8) at project.c:126
126         int result = num_1 * num_2 ;
(gdb) print result
$1 = 2000060248
(gdb) c
Continuing.

Breakpoint 2, multiplier (x=0x61fdf0 "123", y=0x61fd8c "53", base=8) at project.c:131
131         printf( "The result is : %s \n" , new_string ) ;
(gdb) print result
$2 = 3569
```