PP END-SEM PROJECT REPORT

Objectives

- We learned to develop programs for complex real world problems.
- We learned to apply good programming practices in their code like comments and indentation.
- We learned to utilize Debugger and its tools like gdb/gnu for error handling.
- We learned to demonstrate configuration and usage of different software tools used in industry.

Function Description

Function 1 - Arithmetic_Progression

- Arguements: 1 arguement given of datatype int, that is number of terms.
- Input: Takes first term and common difference of AP as input inside the function.
- Return Type: void
- Output: It prints AP using entered first term, common difference and number of terms.

Function 2 - Geometric_Progression

- Arguements: 1 arguement given of datatype int, that is number of terms.
- Input: Takes first term and common ratio of GP as input inside the function.
- Return Type: void
- Output: It prints GP using entered first term, common ratio and number of terms.

Function 3 - Fibonacci_Series

- Define: A term is sum of its previous two terms taking first two terms as 1.
- Arguements: 1 arguement given of datatype int, that is number of terms.
- Return Type: void
- Output: It prints Fibonacci series upto number of terms passed in function.

Function 4 - Hailstone_Sequence

- Define: If a term is n, if n is even then next term = n/2 and if n is odd next term will be 3*n-1.
- Arguements: 1 arguement given of datatype int, that is number of terms.
- Input: It takes first term of the sequence as input inside the function..
- Return Type: void
- Output: It prints Hailstone Sequence upto number of terms passed in function.

Function 5 - Triangular_numbers

- Define: A triangular number are those numbers that counts objects arranged in a equilateral triangle..
- Arguements: 1 arguement given of datatype int, that is number of terms.
- Return Type: void
- Output: It prints triangular numbers upto number of terms passed in function.

Function 6 - Palindromic_numbers

- Define: Numbers which are same if we read it either from the start or from the end are palindromic numbers.
- Arguements: 2 arguements given of datatype int, that is start and end inclusive of the range in which we want to check palindromic numbers.
- Return Type: void
- Output: It prints all the palindromic numbers between the range passed in function.

Function 7 - Prime_numbers

- Define: Numbers which do not have any perfect divisors are prime numbers.
- Arguements: 2 arguements given of datatype int, that is start and end inclusive of the range in which we want to check palindromic numbers.
- Return Type: void
- Output: It prints all the prime numbers between the range passed in function.

Function 8 - Armtrong_numbers

- Define: A three digit integer is integer such that the sum of the cubes of its digit is equal to the number itself.
- Arguements: 2 arguements given of datatype int(only for 3 digit numbers), that is start and end inclusive of the range in which we want to check palindromic numbers.
- Return Type: void
- Output: It prints all the armstrong numbers between the range passed in function.

Function 9 - Sum_of_Sq_Cube

- Define: This function prints the sum of square and cube of natural numbers.
- Arguements: 1 arguement given of datatype int, that is number of terms.
- Return Type: void
- Output: It prints sum of sqaure and cube of natural numbers upto number of terms passed in function.

Function 10 - Power_Series

- Define: This function prints the powers of entered number from 1 to 10.
- Arguements: 1 arguement given of datatype int, that is number whose powers have to printed.
- Return Type: void
- Output: It prints powers from 1 to 10 of the number passed in the function.

Codes

Code screenshots and Output in JAVA

```
import java.util.Scanner;
class java
{

//mini project coantaining 10 functions
 static void Arithmetic_Progression(int n)                              // n is number of terms to be printed;
{Scanner sc=new Scanner(System.in);
    int first_term, common_diff;
    System.out.println ("Enter first term of Arithmetic Progression");
     first_term = sc.nextInt();                                                    //for first term of AP
     System.out.println("Enter common difference of Arithmetic Progression");
     common_diff= sc.nextInt();
     System.out.println("AP");                                                  //for common difference
    for (int term_number = 1; term_number <= n; term_number++){                  //loop for printing AP
        int term = first_term + (term_number - 1) * common_diff;
        System.out.print(term + " ");
    }
    System.out.println();
}

static void Geometric_Progression(int n)                                        // n is number of terms to be printed;
{Scanner sc=new Scanner(System.in);
    int first_term, common_ratio;
    System.out.println("Enter first term of Geometric Progression");
     first_term= sc.nextInt();                                                  //first term of gp
    System.out.println( "Enter common ratio of Geometric Progression");
     common_ratio=sc.nextInt();                                                 //common ratio of GP
    System.out.println("Required Geometric Progression is: ");
    int term = first_term;
    System.out.println("GP");
    for (int term_number = 1; term_number <= n; term_number++){                  //loop for printing GP
        System.out.print(term + " ");
        term = term * common_ratio;
    }
    System.out.println();
}

static void Fibonacci_Series(int n)                                             //n is terms to be printed in fibonacci series
{   System.out.print( "Fibonacci series" );
    int term1 = 1, term2 = 1;                                                  //first two terms of fibonacci series
    System.out.println( term1 + " " + term2 + " ");
    for (int term_number = 3; term_number <= n; term_number++){                //loop for calculating next term in fibonacci series
        int temp = term1 + term2;
        term1 = term2;
```

```java
static void Fibonacci_Series(int n)
{   System.out.print( "Fibonacci series" );
    int term1 = 1, term2 = 1;
    System.out.println( term1 + " " + term2 + " ");
    for (int term_number = 3; term_number <= n; term_number++){
        int temp = term1 + term2;
        term1 = term2;
        term2 = temp;
        System.out.print( term2 + " ");
    }
    System.out.println();
}

static void Hailstone_sequence(int n)
{System.out.print("hailstone series");
    Scanner sc=new Scanner(System.in);
    int first_term;
    System.out.println("\nEnter a number to start hailstone sequence: \n");
     first_term=sc.nextInt();
    int term = first_term;
    for (int term_number = 1; term_number <= n; term_number++){
                            //loop for calculating the hailstone series
        System.out.print(term + " ");
        if (term % 2 == 0){
            term = term / 2;
        }else{
            term = 3 * term + 1;
        }
    }
    System.out.println();
}

static void Triangular_numbers(int n)
{  System.out.print("triangular number");
    for (int term = 1; term <= n; term++){                                   //loop
        int triangular = term * (term + 1) / 2;

        System.out.print( triangular + " ");
    }
    System.out.println();
}

static void Palindromic_numbers(int start_check, int end_check)
{System.out.print("Palindrome series");
```

guptasumit2034@LAPTOP-Q7D19B6K: ~/profiling/pp_mini_project

GNU nano 4.8

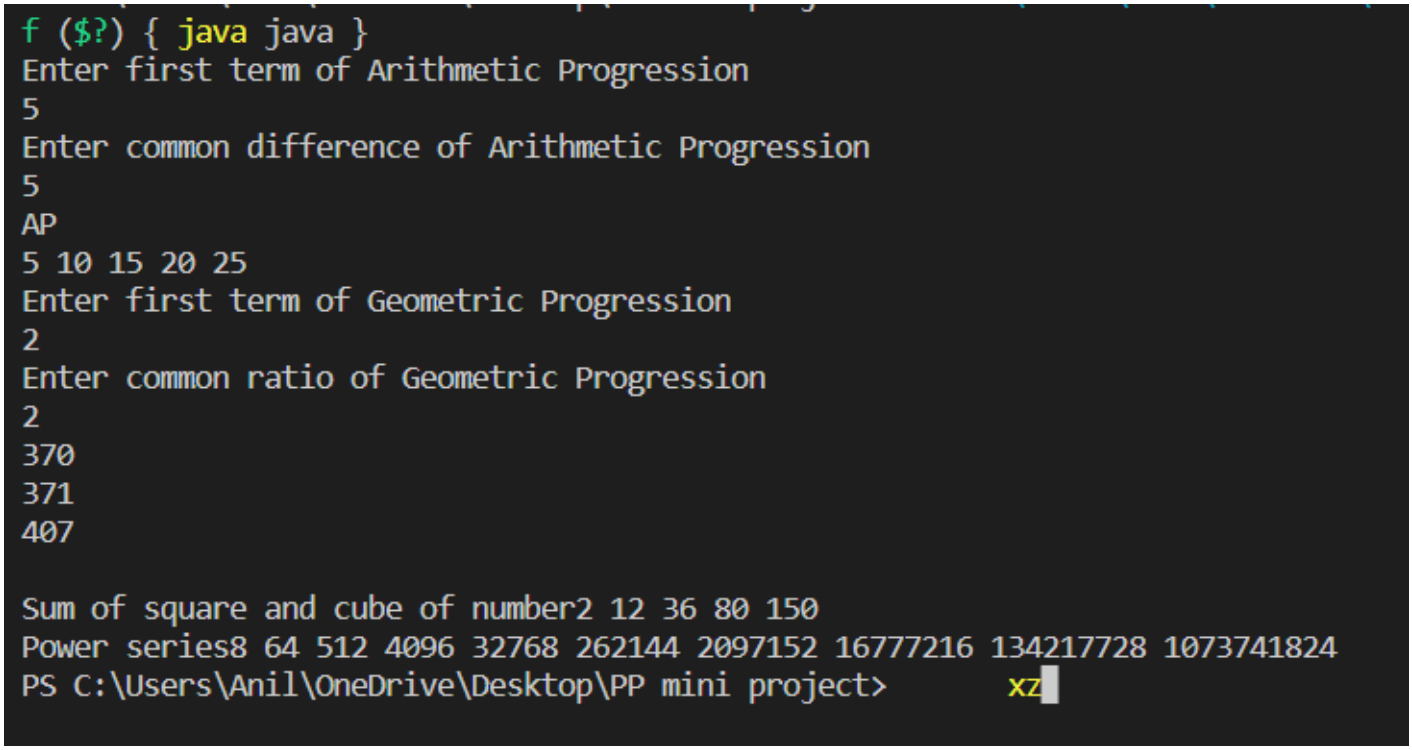
```
static void Palindromic_numbers(int start_check, int end_check)
{System.out.print("Palindrome series");
    for (int number = start_check; number <= end_check; number++){
        int temp = number;
        int rev = 0;
        while (temp!=0)
        {
            rev = rev * 10 + temp % 10;
            temp = temp / 10;
        }

        if (number == rev)
        {

            System.out.print (number + " ");
        }
    }
    System.out.println();
}

static void Prime_numbers(int start_check, int end_check)
{System.out.print("Prime number series");
    for (int number = start_check; number <= end_check; number++){
        int flag = 1;
        for (int divisor = 2; divisor <= number / 2; divisor++){
            if (number % divisor == 0){
                flag = 0;
            }
        }

        if(flag == 1){
            System.out.print(number+" ");
        }
    }
    System.out.println();
}

static void Armstrong_numbers(int start_check, int end_check)

{
    System.out.print("Armstrong");
    for (int number = start_check; number <= end_check; number++){
        int temp=number;
        int sum =0;
```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell

guptasumit2034@LAPTOP-Q7D19B6K: ~/profilling/pp_mini_project

GNU nano 4.8

```
static void Armstrong_numbers(int start_check, int end_check)

{
    System.out.print("Armstrong");
    for (int number = start_check; number <= end_check; number++){
        int temp=number;
        int sum =0;
        while(temp!=0){
            int digit=temp%10;
            sum=sum+digit*digit*digit;
            temp=temp/10;
        }

        if(number == sum){
            System.out.println(number+" ");
        }
    }
    System.out.println();
}

static void Sum_of_Sq_Cube(int n)
{  System.out.print("Sum of square and cube of number");
    for(int term_number=1;term_number<=n;term_number++){
        int sum = term_number*term_number*(term_number+1);
        System.out.print(sum+" ");
    }
    System.out.println();
}

static void Power_series(int n)
{ System.out.print("Power series");
     int answer=1;
    for(int power=1;power<=10;power++){
        answer=answer*n;
        System.out.print(answer+" ");
    }
    System.out.println();
}
```



Codes

Code screenshots and Output in C++

guptasumit2034@LAPTOP-Q7D19B6K: ~

GNU nano 4.8 cpp.cpp

```cpp
#include <iostream>

using namespace std;
//mini project coantaining 10 functions
void Arithmetic_Progression(int n)                          // n is number of terms to be printed;
{
    int first_term, common_diff;
    cout << "Enter first term of Arithmetic Progression\n";
    cin >> first_term;                                                  //for first term of AP
    cout << "Enter common difference of Arithmetic Progression\n";
    cin >> common_diff;                                                 //for common difference
    cout << "Required Arithmetic Progression is: \n";
    for (int term_number = 1; term_number <= n; term_number++){          //loop for printing AP
        int term = first_term + (term_number - 1) * common_diff;
        cout << term << " ";
    }
    cout << "\n\n";
}

void Geometric_Progression(int n)                                    // n is number of terms to be printed;
{
    int first_term, common_ratio;
    cout << "Enter first term of Geometric Progression\n";
    cin >> first_term;                                                  //first term of gp
    cout << "Enter common ratio of Geometric Progression\n";
    cin >> common_ratio;                                                //common ratio of GP
    cout << "Required Geometric Progression is: \n";
    int term = first_term;
    for (int term_number = 1; term_number <= n; term_number++){          //loop for printing GP
        cout << term << " ";
        term = term * common_ratio;
    }
    cout << "\n\n";
}

void Fibonacci_Series(int n)                                        //n is terms to be printed in fibonacci series
{
    int term1 = 1, term2 = 1;                                       //first two terms of fibonacci series
    cout << term1 << " " << term2 << " ";
    for (int term_number = 3; term_number <= n; term_number++){     //loop for calculating next term in fibonacci series
        int temp = term1 + term2;
        term1 = term2;
        term2 = temp;
        cout << term2 << " ";
    }
```

guptasumit2034@LAPTOP-Q7D19B6K: ~

GNU nano 4.8

```
void Hailstone_sequence(int n)
{
    int first_term;
    cout << "\nEnter a number to start hailstone sequence: \n";
    cin >> first_term;
    int term = first_term;
    for (int term_number = 1; term_number <= n; term_number++){
        cout << term << " ";
        if (term % 2 == 0){
            term = term / 2;
        }else{
            term = 3 * term + 1;
        }
    }
    cout << "\n";
}

void Triangular_numbers(int n)                                          //number
{
    for (int term = 1; term <= n; term++){                               //loop
        int triangular = term * (term + 1) / 2;
        cout << triangular << " ";
    }
    cout << "\n";
}

void Palindromic_numbers(int start_check, int end_check)
{
    for (int number = start_check; number <= end_check; number++){
        int temp = number;
        int rev = 0;
        while (temp)
        {
            rev = rev * 10 + temp % 10;
            temp = temp / 10;
        }
        if (number == rev)
        {
            cout << number << " ";
        }
    }
    cout << "\n";
}
```

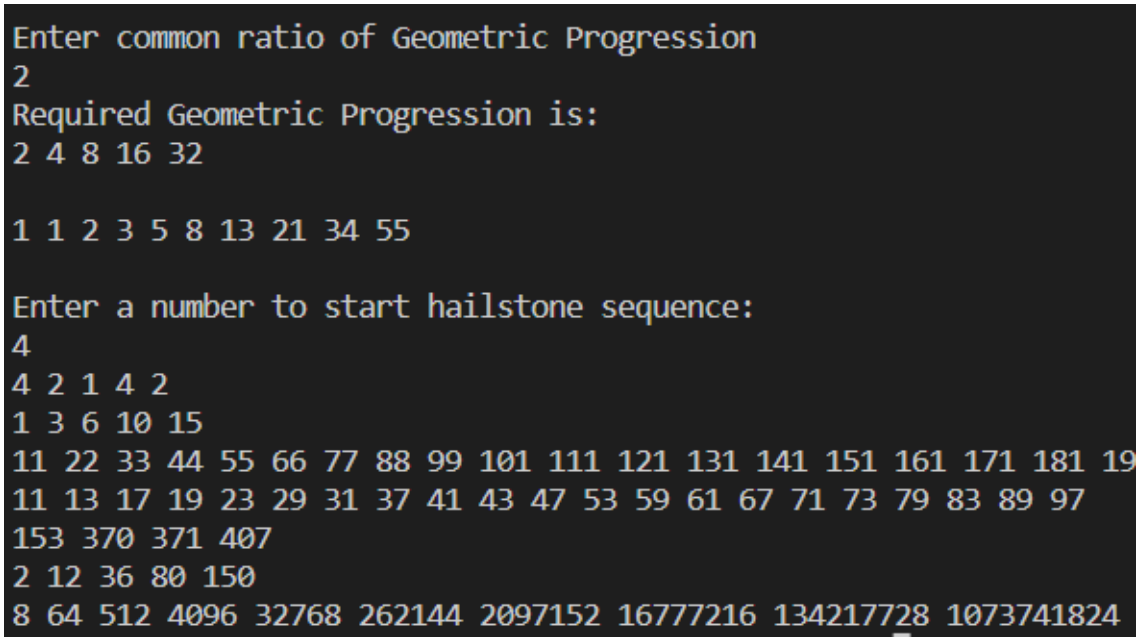
guptasumit2034@LAPTOP-Q7D19B6K: ~

GNU nano 4.8

```
void Prime_numbers(int start_check, int end_check)
{
    for (int number = start_check; number <= end_check; number++){
        int flag = 1;
        for (int divisor = 2; divisor <= number / 2; divisor++){
            if (number % divisor == 0){
                flag = 0;
            }
        }
        if(flag == 1){
            cout<<number<<" ";
        }
    }
    cout << "\n";
}

void Armstrong_numbers(int start_check, int end_check)

{
    for (int number = start_check; number <= end_check; number++){
        int temp=number;
        int sum =0;
        while(temp){
            int digit=temp%10;
            sum=sum+digit*digit*digit;
            temp=temp/10;
        }
        if(number == sum){
            cout<<number<<" ";
        }
    }
    cout << "\n";
}

void Sum_of_Sq_Cube(int n)
{
    for(int term_number=1;term_number<=n;term_number++){
        int sum = term_number*term_number*(term_number+1);
        cout<<sum<<" ";
    }
    cout<<"\n";
}

void Power_series(int n)
```

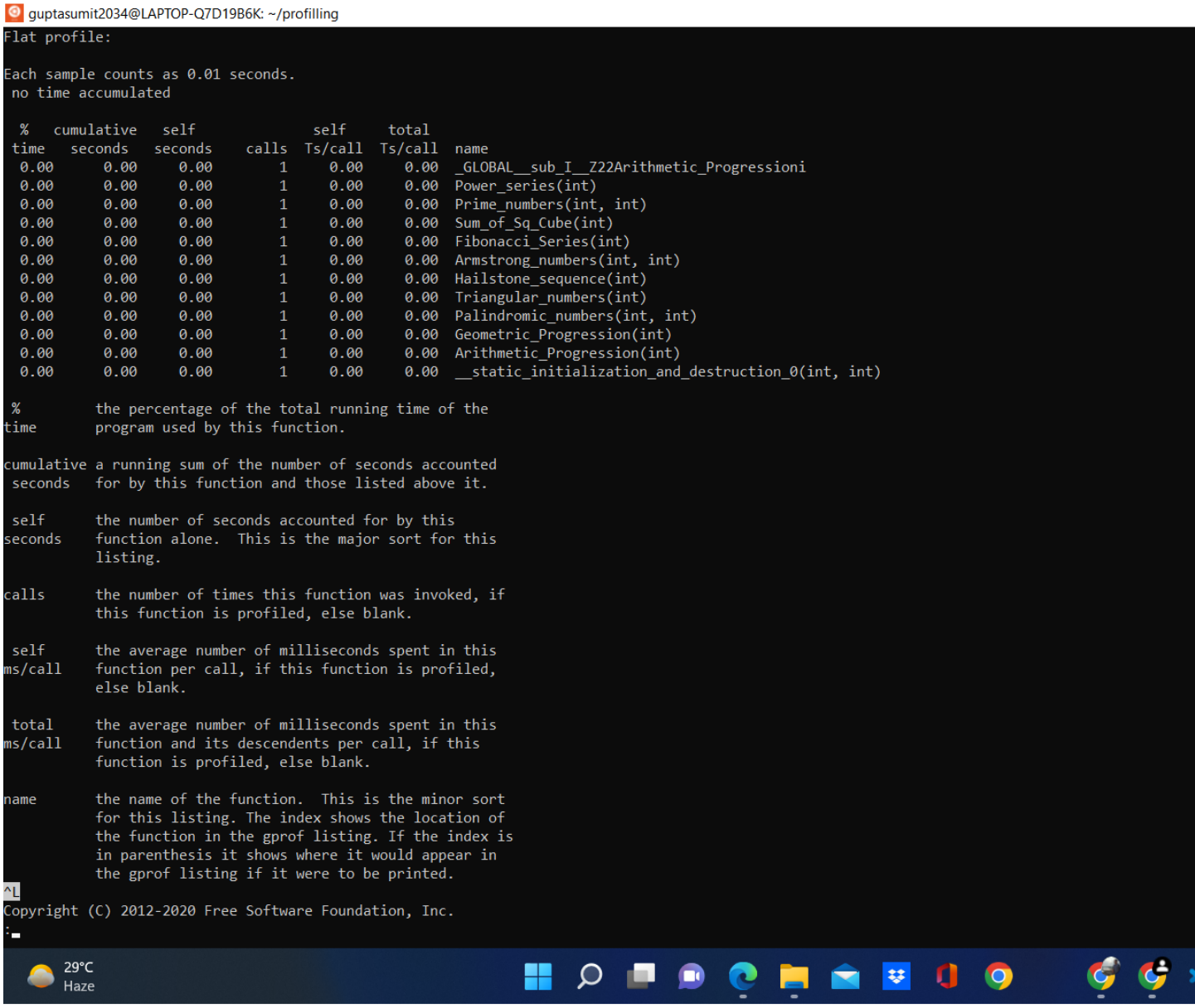
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify

```
void Power_series(int n)
{
    long long int answer=1;
    for(int power=1;power<=10;power++){
        answer=answer*n;
        cout<<answer<<" ";
    }
    cout<<"\n";
}

int main()
{
    Arithmetic_Progression(5);
    Geometric_Progression(5);
    Fibonacci_Series(10);
    Hailstone_sequence(5);
    Triangular_numbers(5);
    Palindromic_numbers(10, 200);
    Prime_numbers(10,100);
    Armstrong_numbers(100,999);
    Sum_of_Sq_Cube(5);
    Power_series(8);
}
```



Profile Report

Profile Report Screenshots


guptasumit2034@LAPTOP-Q7D19B6K: ~/profilling
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 1 0.00 0.00 _GLOBAL__sub_I__Z22Arithmetic_Progressioni
0.00 0.00 0.00 1 0.00 0.00 Power_series(int)
0.00 0.00 0.00 1 0.00 0.00 Prime_numbers(int, int)
0.00 0.00 0.00 1 0.00 0.00 Sum_of_Sq_Cube(int)
0.00 0.00 0.00 1 0.00 0.00 Fibonacci_Series(int)
0.00 0.00 0.00 1 0.00 0.00 Armstrong_numbers(int, int)
0.00 0.00 0.00 1 0.00 0.00 Hailstone_sequence(int)
0.00 0.00 0.00 1 0.00 0.00 Triangular_numbers(int)
0.00 0.00 0.00 1 0.00 0.00 Palindromic_numbers(int, int)
0.00 0.00 0.00 1 0.00 0.00 Geometric_Progression(int)
0.00 0.00 0.00 1 0.00 0.00 Arithmetic_Progression(int)
0.00 0.00 0.00 1 0.00 0.00 __static_initialization_and_destruction_0(int, int)
% 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.
calls the number of times this function was invoked, if
this function is profiled, else blank.
self the average number of milliseconds spent in this
ms/call function per call, if this function is profiled,
else blank.
total the average number of milliseconds spent in this
ms/call function and its descendents per call, if this
function is profiled, else blank.
name the name of the function. This is the minor sort
for this listing. The index shows the location of
the function in the gprof listing. If the index is
in parenthesis it shows where it would appear in
the gprof listing if it were to be printed.
^L
Copyright (C) 2012-2020 Free Software Foundation, Inc.
:
29°C
Haze

guptasumit2034@LAPTOP-Q7D19B6K: ~/profilling

```
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.
^L
                     Call graph (explanation follows)


granularity: each sample hit covers 2 byte(s) no time propagated

index % time    self  children    called     name
                0.00    0.00       1/1           __libc_csu_init [23]
[8]      0.0    0.00    0.00       1         _GLOBAL__sub_I__Z22Arithmetic_Progress
                0.00    0.00       1/1           __static_initialization_and_destru
-----------------------------------------------
                0.00    0.00       1/1           main [6]
[9]      0.0    0.00    0.00       1         Power_series(int) [9]
-----------------------------------------------
                0.00    0.00       1/1           main [6]
[10]     0.0    0.00    0.00       1         Prime_numbers(int, int) [10]
-----------------------------------------------
                0.00    0.00       1/1           main [6]
[11]     0.0    0.00    0.00       1         Sum_of_Sq_Cube(int) [11]
-----------------------------------------------
                0.00    0.00       1/1           main [6]
[12]     0.0    0.00    0.00       1         Fibonacci_Series(int) [12]
-----------------------------------------------
                0.00    0.00       1/1           main [6]
[13]     0.0    0.00    0.00       1         Armstrong_numbers(int, int) [13]
-----------------------------------------------
                0.00    0.00       1/1           main [6]
[14]     0.0    0.00    0.00       1         Hailstone_sequence(int) [14]
-----------------------------------------------
                0.00    0.00       1/1           main [6]
[15]     0.0    0.00    0.00       1         Triangular_numbers(int) [15]
-----------------------------------------------
                0.00    0.00       1/1           main [6]
[16]     0.0    0.00    0.00       1         Palindromic_numbers(int, int) [16]
-----------------------------------------------
                0.00    0.00       1/1           main [6]
[17]     0.0    0.00    0.00       1         Geometric_Progression(int) [17]
-----------------------------------------------
                0.00    0.00       1/1           main [6]
[18]     0.0    0.00    0.00       1         Arithmetic_Progression(int) [18]
-----------------------------------------------
                0.00    0.00       1/1           _GLOBAL__sub_I__Z22Arithmetic_Prog
[19]     0.0    0.00    0.00       1         __static_initialization_and_destructio
-----------------------------------------------
```

guptasumit2034@LAPTOP-Q7D19B6K: ~/profilling

```
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
               in this function and its children.  Note that due to
               different viewpoints, functions excluded by options, etc,
               these numbers will NOT add up to 100%.

    self       This is the total amount of time spent in this function.

    children   This is the total amount of time propagated into this
               function by its children.

    called     This is the number of times the function was called.
               If the function called itself recursively, the number
               only includes non-recursive calls, and is followed by
               a `+' and the number of recursive calls.

    name       The name of the current function.  The index number is
               printed after it.  If the function is a member of a
               cycle, the cycle number is printed between the
               function's name and the index number.


For the function's parents, the fields have the following meanings:

    self       This is the amount of time that was propagated directly
               from the function into this parent.

    children   This is the amount of time that was propagated from
               the function's children into this parent.

    called     This is the number of times this parent called the
               function `/' the total number of times the function
               was called.  Recursive calls to the function are not
               included in the number after the `/'.

    name       This is the name of the parent.  The parent's index
```

guptasumit2034@LAPTOP-Q7D19B6K: ~/profilling

```
    name        This is the name of the parent.  The parent's index
                number is printed after it.  If the parent is a
                member of a cycle, the cycle number is printed between
                the name and the index number.

If the parents of the function cannot be determined, the word
`<spontaneous>' is printed in the `name' field, and all the other
fields are blank.

For the function's children, the fields have the following meanings:

    self        This is the amount of time that was propagated directly
                from the child into the function.

    children    This is the amount of time that was propagated from the
                child's children to the function.

    called      This is the number of times the function called
                this child `/' the total number of times the child
                was called.  Recursive calls by the child are not
                listed in the number after the `/'.

    name        This is the name of the child.  The child's index
                number is printed after it.  If the child is a
                member of a cycle, the cycle number is printed
                between the name and the index number.

If there are any cycles (circles) in the call graph, there is an
entry for the cycle-as-a-whole.  This entry shows who called the
cycle (as parents) and the members of the cycle (as children.)
The `+' recursive calls entry shows the number of function calls that
were internal to the cycle, and the calls entry for each member shows,
for that member, how many times it was called from other members of
the cycle.
^L
Copyright (C) 2012-2020 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.
^L
Index by function name

   [8] _GLOBAL__sub_I__Z22Arithmetic_Progressioni [12] Fibonacci_Series(int) [16] P
   [9] Power_series(int)      [13] Armstrong_numbers(int, int) [17] Geometric_Progr
  [10] Prime_numbers(int, int) [14] Hailstone_sequence(int) [18] Arithmetic_Progres
  [11] Sum_of_Sq_Cube(int)    [15] Triangular_numbers(int) [19] __static_initializa
```

```
Reading symbols from a.out...
(gdb) break 5
Breakpoint 1 at 0x1209: file main.cpp, line 6.
(gdb) break 12
Breakpoint 2 at 0x1273: file main.cpp, line 12.
(gdb) break 148
Breakpoint 3 at 0x191e: file main.cpp, line 148.
(gdb) run
Starting program: /home/a.out

Breakpoint 3, main () at main.cpp:148
148         Arithmetic_Progression(5);
(gdb) n

Breakpoint 1, Arithmetic_Progression (n=21845) at main.cpp:6
6       {
(gdb) n
8           cout << "Enter first term of Arithmetic Progression\n";
(gdb) n
Enter first term of Arithmetic Progression
9           cin >> first_term;                                                  //for first term of AP
(gdb) n
4
10          cout << "Enter common difference of Arithmetic Progression\n";
(gdb) n
Enter common difference of Arithmetic Progression
11          cin >> common_diff;                                                 //for common difference
(gdb) n
5

Breakpoint 2, Arithmetic_Progression (n=5) at main.cpp:12
12          cout << "Required Arithmetic Progression is: \n";
(gdb) n
Required Arithmetic Progression is:
13          for (int term_number = 1; term_number <= n; term_number++){          //loop for printing AP
(gdb) c
Continuing.
```

```
(gdb) n
5

Breakpoint 2, Arithmetic_Progression (n=5) at main.cpp:12
12          cout << "Required Arithmetic Progression is: \n";
(gdb) n
Required Arithmetic Progression is:
13          for (int term_number = 1; term_number <= n; term_number+
(gdb) c
Continuing.

Breakpoint 4, Arithmetic_Progression (n=5) at main.cpp:17
17          cout << "\n\n";
(gdb) c
Continuing.
4 9 14 19 24

Enter first term of Geometric Progression
1
Enter common ratio of Geometric Progression
2
Required Geometric Progression is:
1 2 4 8 16

1 1 2 3 5 8 13 21 34 55

Enter a number to start hailstone sequence:
5
5 16 8 4 2
1 3 6 10 15
11 22 33 44 55 66 77 88 99 101 111 121 131 141 151 161 171 181 191
11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
153 370 371 407
2 12 36 80 150
8 64 512 4096 32768 262144 2097152 16777216 134217728 1073741824
[Inferior 1 (process 166) exited normally]
(gdb)
```