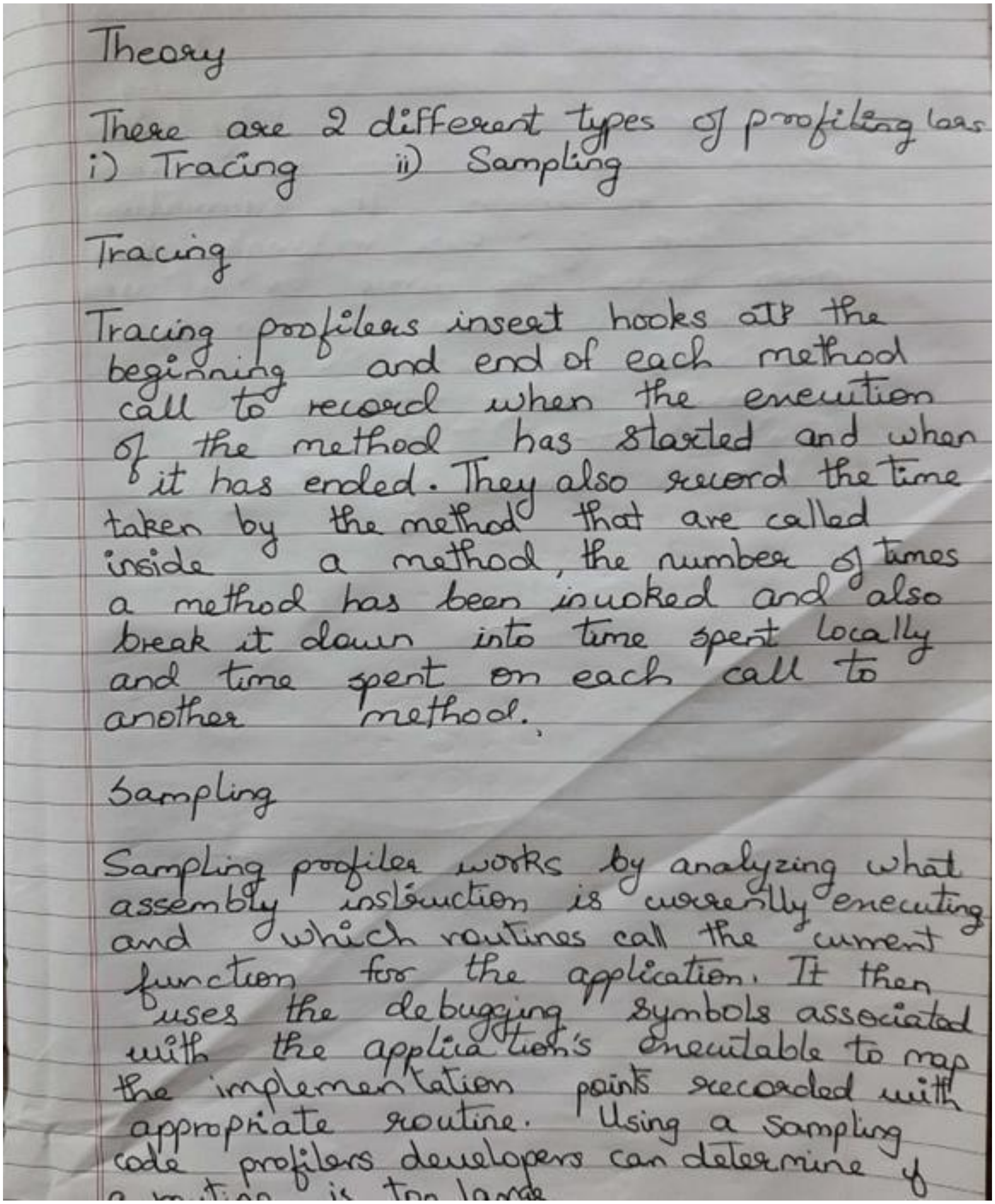
Parallel Programming Lab 7: Profiling

Name: Raghav Maheshwari	Batch: A4
Roll no: PA53	Panel: A

Aim: Using gprof utility analyze the code written in 1 or 2 **without parallelism.**

Objective: To profile a code to find hotspots

Theory:



Theory

There are 2 different types of profiling tools
i) Tracing ii) Sampling

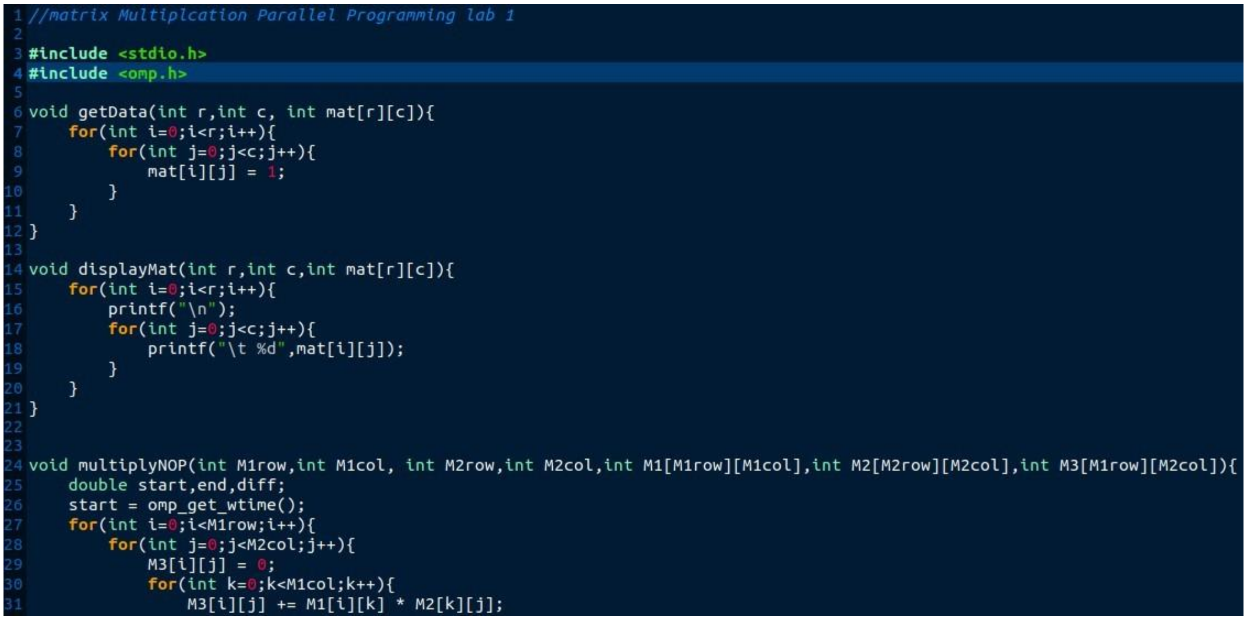
Tracing

Tracing profilers insert hooks at the beginning and end of each method call to record when the execution of the method has started and when it has ended. They also record the time taken by the method that are called inside a method, the number of times a method has been invoked and also break it down into time spent locally and time spent on each call to another method.

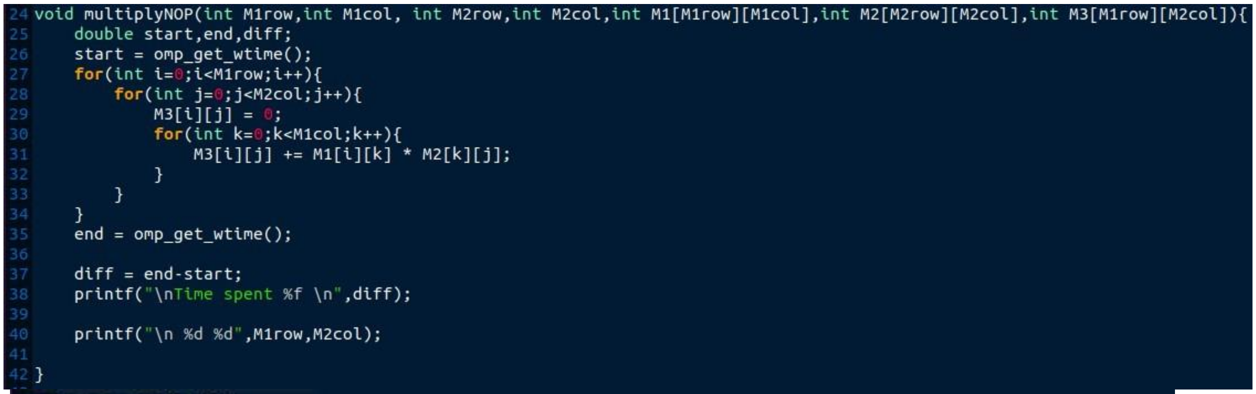
Sampling

Sampling profiler works by analyzing what assembly instruction is currently executing and which routines call the current function for the application. It then uses the debugging symbols associated with the application's executable to map the implementation points recorded with appropriate routine. Using a sampling code profilers developers can determine if a routine is too large

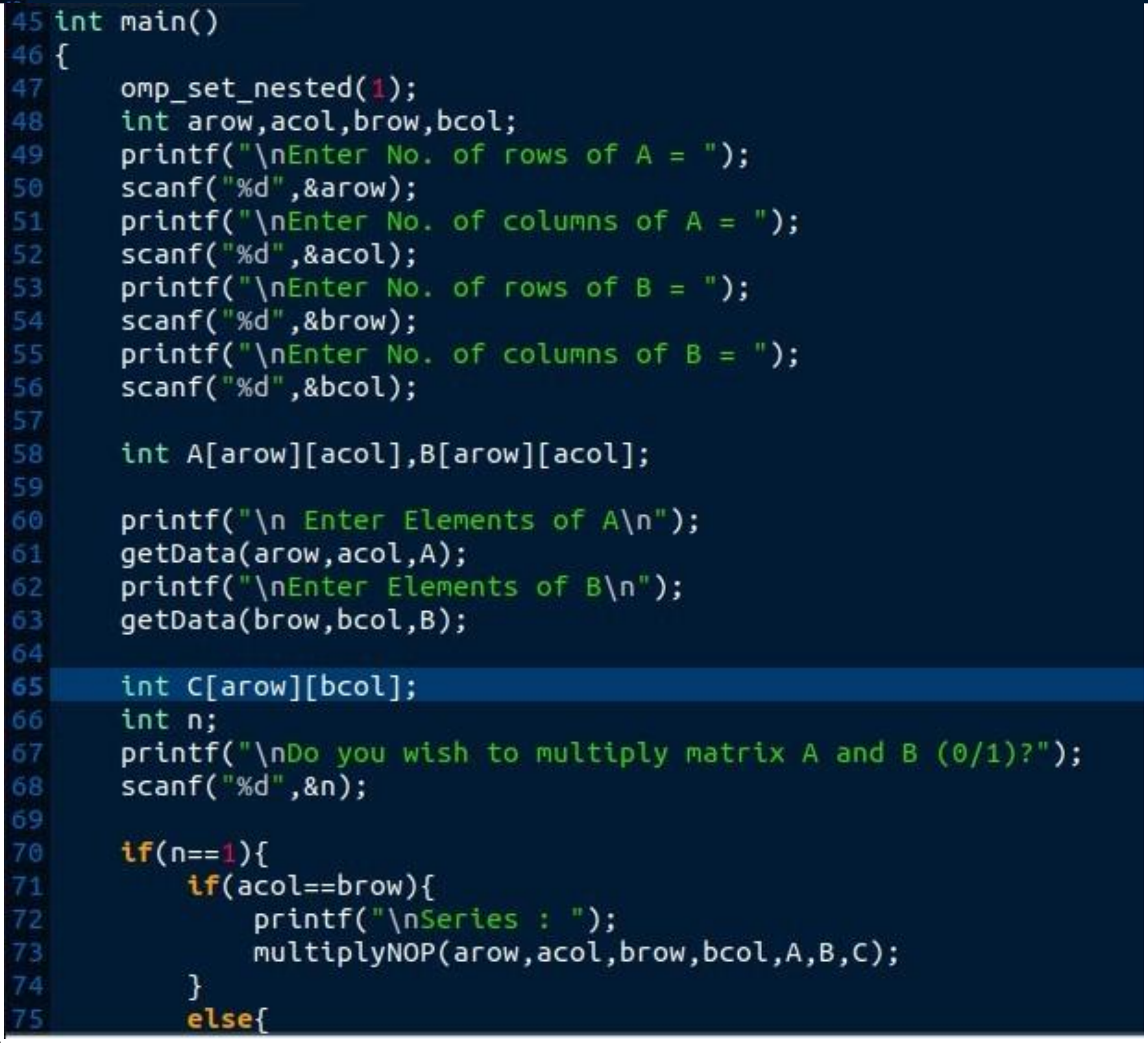
Code:



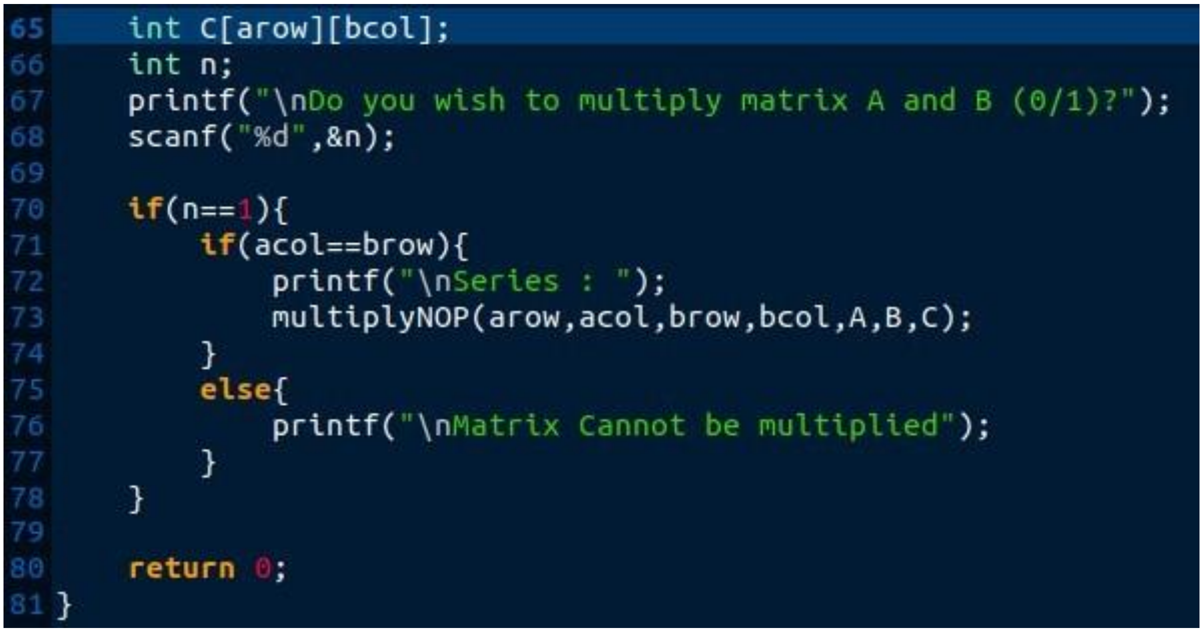
```c
//matrix Multiplcation Parallel Programming lab 1

#include <stdio.h>
#include <omp.h>

void getData(int r,int c, int mat[r][c]){
    for(int i=0;i<r;i++){
        for(int j=0;j<c;j++){
            mat[i][j] = 1;
        }
    }
}

void displayMat(int r,int c,int mat[r][c]){
    for(int i=0;i<r;i++){
        printf("\n");
        for(int j=0;j<c;j++){
            printf("\t %d",mat[i][j]);
        }
    }
}


void multiplyNOP(int M1row,int M1col, int M2row,int M2col,int M1[M1row][M1col],int M2[M2row][M2col],int M3[M1row][M2col]){
    double start,end,diff;
    start = omp_get_wtime();
    for(int i=0;i<M1row;i++){
        for(int j=0;j<M2col;j++){
            M3[i][j] = 0;
            for(int k=0;k<M1col;k++){
                M3[i][j] += M1[i][k] * M2[k][j];
```



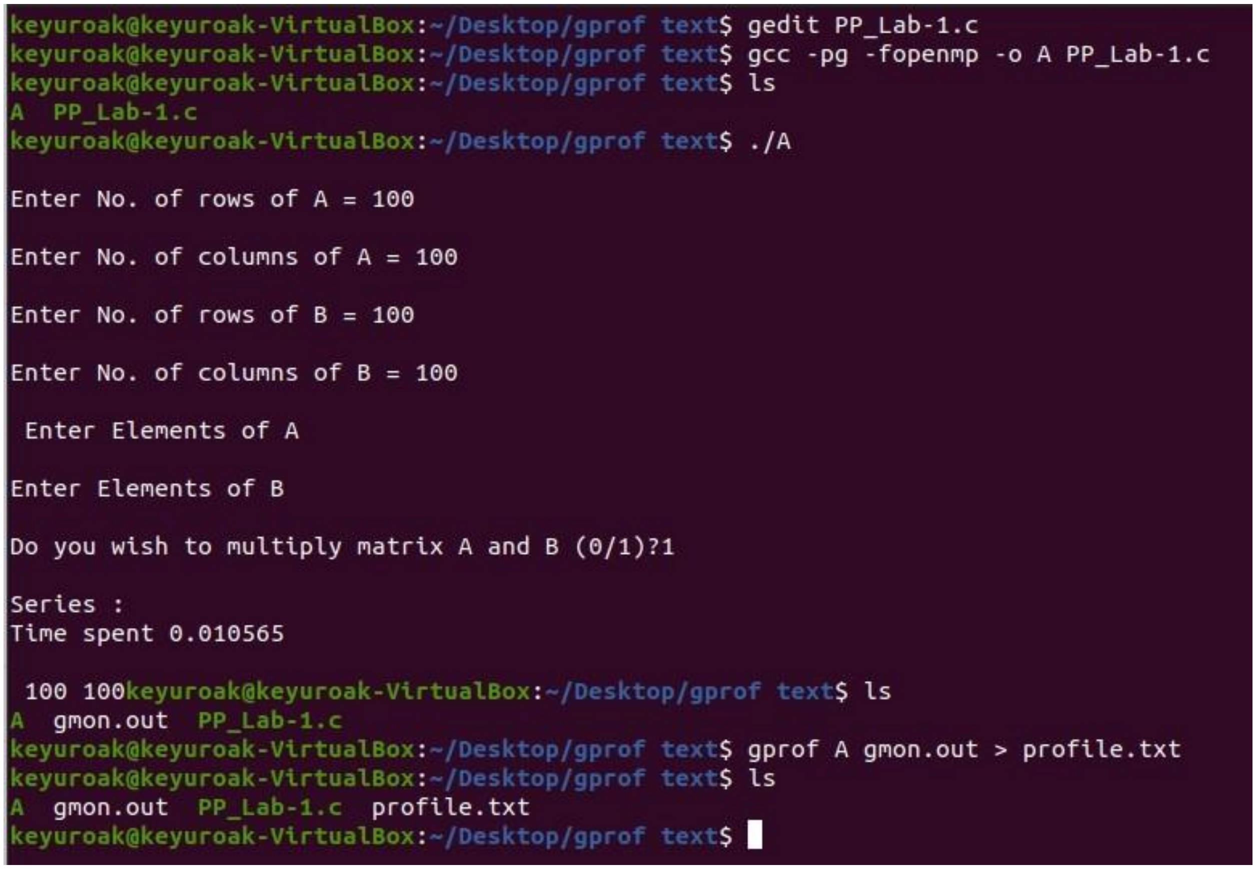
```c
void multiplyNOP(int M1row,int M1col, int M2row,int M2col,int M1[M1row][M1col],int M2[M2row][M2col],int M3[M1row][M2col]){
    double start,end,diff;
    start = omp_get_wtime();
    for(int i=0;i<M1row;i++){
        for(int j=0;j<M2col;j++){
            M3[i][j] = 0;
            for(int k=0;k<M1col;k++){
                M3[i][j] += M1[i][k] * M2[k][j];
            }
        }
    }
    end = omp_get_wtime();

    diff = end-start;
    printf("\nTime spent %f \n",diff);

    printf("\n %d %d",M1row,M2col);

}
```



```c
int main()
{
    omp_set_nested(1);
    int arow,acol,brow,bcol;
    printf("\nEnter No. of rows of A = ");
    scanf("%d",&arow);
    printf("\nEnter No. of columns of A = ");
    scanf("%d",&acol);
    printf("\nEnter No. of rows of B = ");
    scanf("%d",&brow);
    printf("\nEnter No. of columns of B = ");
    scanf("%d",&bcol);

    int A[arow][acol],B[arow][acol];

    printf("\n Enter Elements of A\n");
    getData(arow,acol,A);
    printf("\nEnter Elements of B\n");
    getData(brow,bcol,B);

    int C[arow][bcol];
    int n;
    printf("\nDo you wish to multiply matrix A and B (0/1)?");
    scanf("%d",&n);

    if(n==1){
        if(acol==brow){
            printf("\nSeries : ");
            multiplyNOP(arow,acol,brow,bcol,A,B,C);
        }
        else{
```

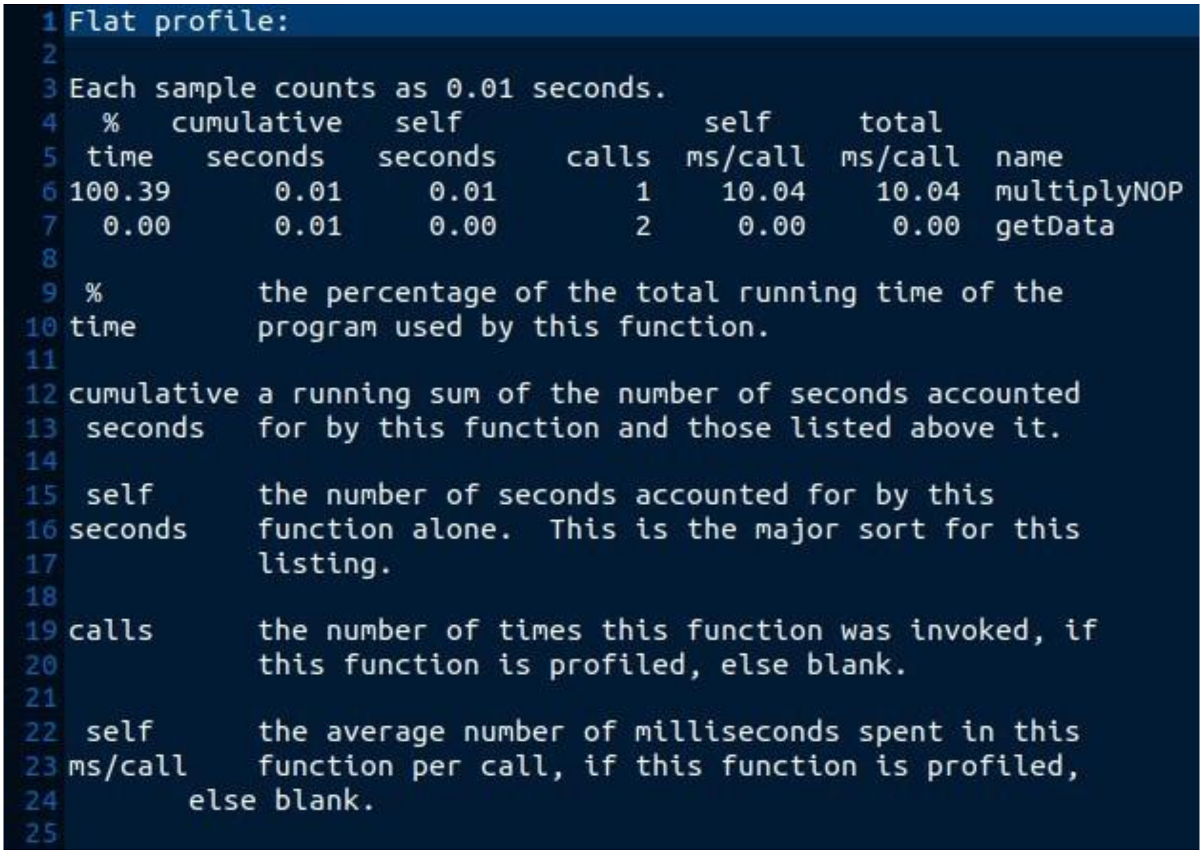


```c
    int C[arow][bcol];
    int n;
    printf("\nDo you wish to multiply matrix A and B (0/1)?");
    scanf("%d",&n);

    if(n==1){
        if(acol==brow){
            printf("\nSeries : ");
            multiplyNOP(arow,acol,brow,bcol,A,B,C);
        }
        else{
            printf("\nMatrix Cannot be multiplied");
        }
    }

    return 0;
}
```

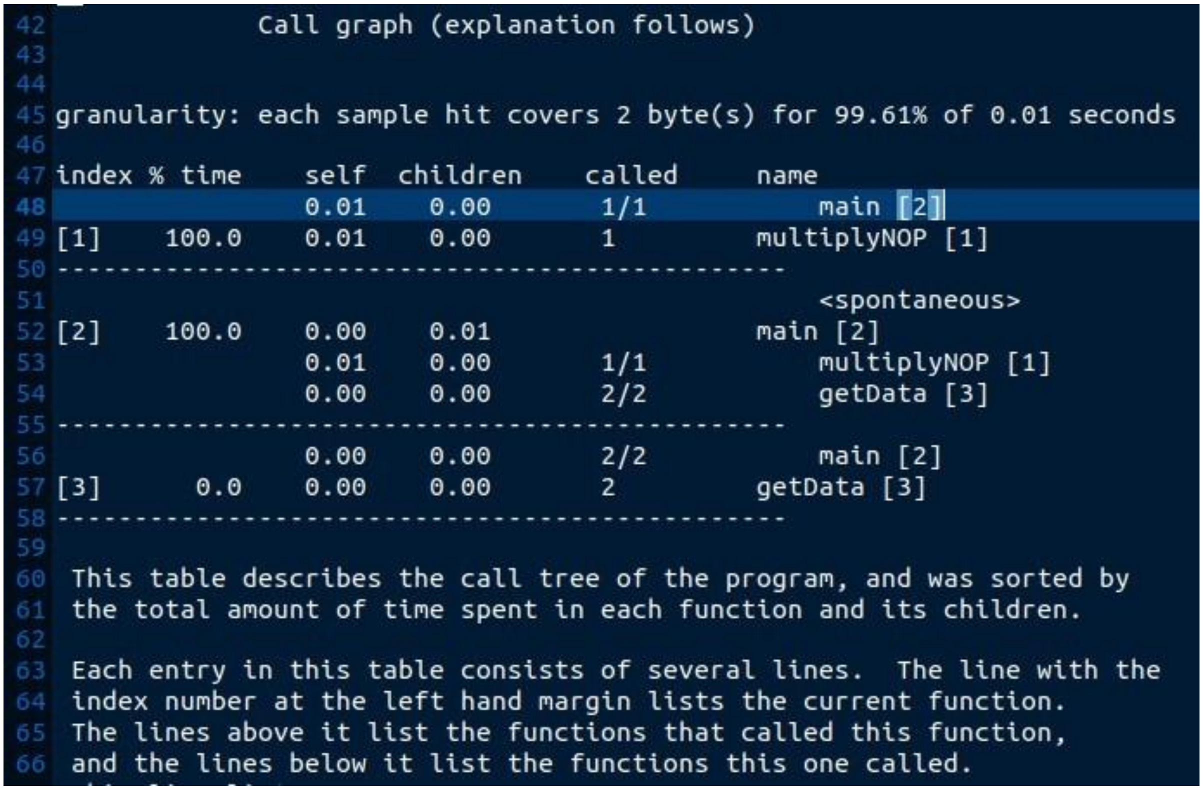


```
keyuroak@keyuroak-VirtualBox:~/Desktop/gprof text$ gedit PP_Lab-1.c
keyuroak@keyuroak-VirtualBox:~/Desktop/gprof text$ gcc -pg -fopenmp -o A PP_Lab-1.c
keyuroak@keyuroak-VirtualBox:~/Desktop/gprof text$ ls
A  PP_Lab-1.c
keyuroak@keyuroak-VirtualBox:~/Desktop/gprof text$ ./A

Enter No. of rows of A = 100

Enter No. of columns of A = 100

Enter No. of rows of B = 100

Enter No. of columns of B = 100

 Enter Elements of A

Enter Elements of B

Do you wish to multiply matrix A and B (0/1)?1

Series :
Time spent 0.010565

 100 100keyuroak@keyuroak-VirtualBox:~/Desktop/gprof text$ ls
A  gmon.out  PP_Lab-1.c
keyuroak@keyuroak-VirtualBox:~/Desktop/gprof text$ gprof A gmon.out > profile.txt
keyuroak@keyuroak-VirtualBox:~/Desktop/gprof text$ ls
A  gmon.out  PP_Lab-1.c  profile.txt
keyuroak@keyuroak-VirtualBox:~/Desktop/gprof text$ 
```

Result:



```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
100.39      0.01     0.01        1    10.04    10.04  multiplyNOP
  0.00      0.01     0.00        2     0.00     0.00  getData

 %         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.
```



```
             Call graph (explanation follows)


granularity: each sample hit covers 2 byte(s) for 99.61% of 0.01 seconds

index % time    self  children    called     name
                0.01    0.00       1/1           main [2]
[1]    100.0    0.01    0.00       1         multiplyNOP [1]
-----------------------------------------------
                                                 <spontaneous>
[2]    100.0    0.00    0.01                 main [2]
                0.01    0.00       1/1           multiplyNOP [1]
                0.00    0.00       2/2           getData [3]
-----------------------------------------------
                0.00    0.00       2/2           main [2]
[3]      0.0    0.00    0.00       2         getData [3]
-----------------------------------------------

 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.
```

Conclusion: To profile a code to find hotspots

FAQ's:

1. What are Sampling profilers

Ans: Sampling profilers are useful tools for performance analysis of programs. A sampling profiler runs every N time units and captures the stack running in each thread at that point. The resulting output is usually some combination of the counts and times of the observed stacks. With frequent samples over a long enough time, the output will reflect how the program itself is spending time.

2. What are Instrumenting profilers

An instrumentation profiler works by inserting code at the start and end of a routine. It identifies crucial checkpoints and inserts code into them to record routine sequences, time, or even variable content. There are two types of instrumentation profilers — source-code modifying profiler and binary profiler.