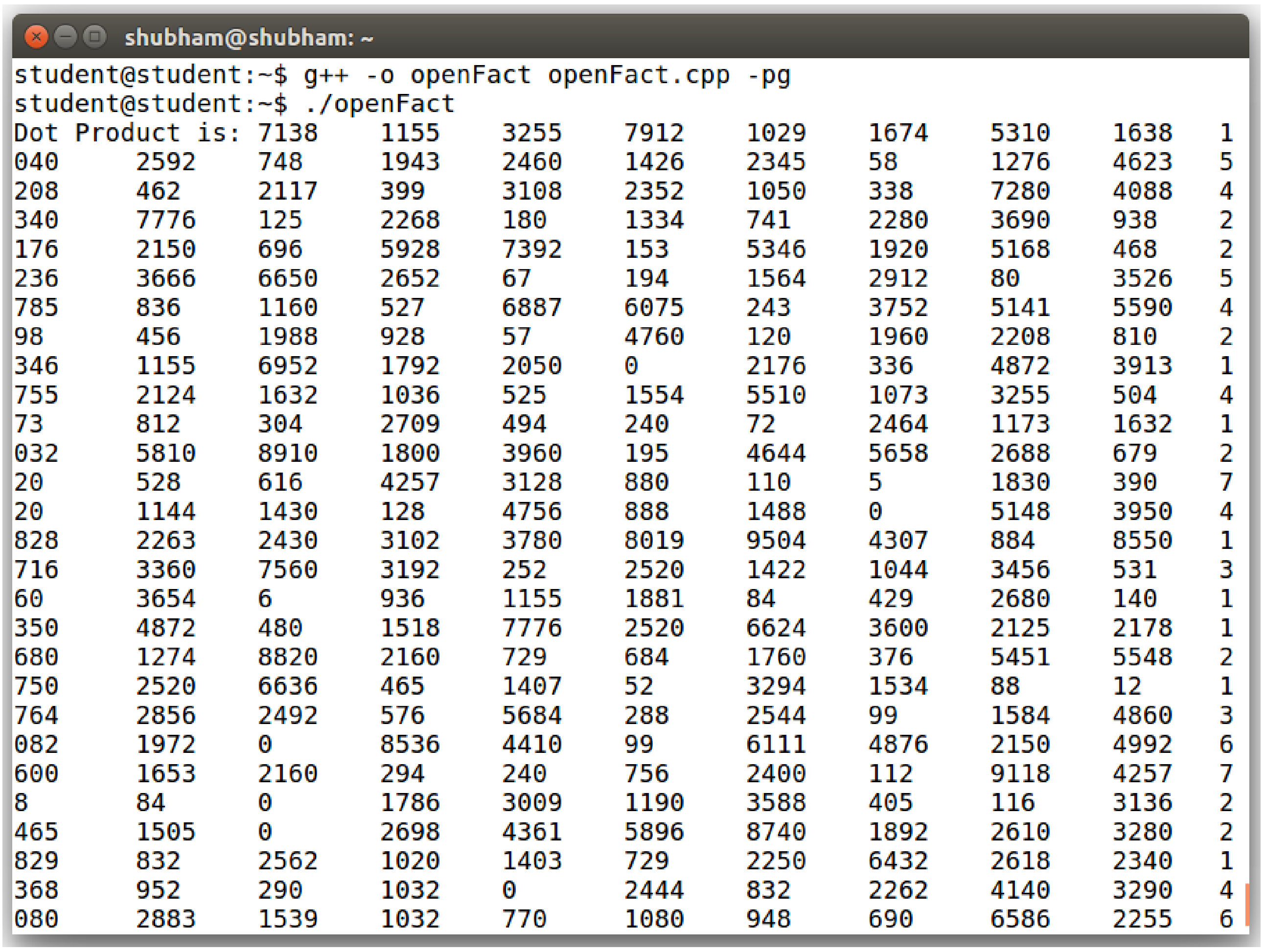
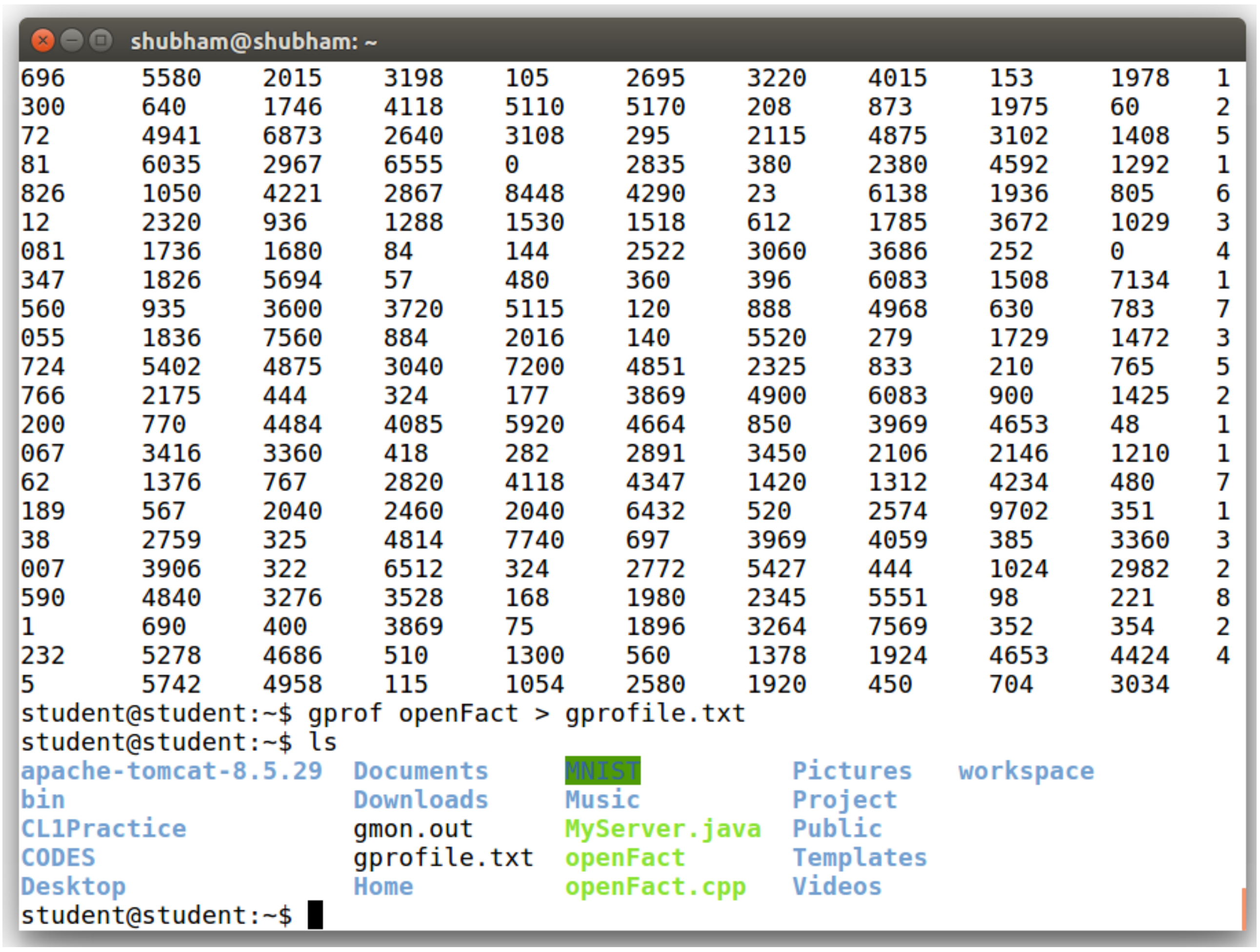
File: openFact.cpp

```
#include <iostream>
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>
#define MAX 1000000
unsigned long long a[MAX],b[MAX],c[MAX];
using namespace std;
void display(void);
void initialize(void) {
    for(int i = 1; i <=MAX; ++i)
    {
     b[i]=rand()%100;
     c[i]=rand()%100;
    }
};

void Multiply(void) {
    for(int i = 1; i <=MAX; ++i) {
     a[i]=b[i]*c[i];
    }

};

int main() {
     initialize();
     display();
     return 0;
}

void display(void) {
     Multiply();
     cout<<"Dot Product is: ";
     for(int i = 1; i <=MAX; ++i)
     cout<<a[i]<<"\t";
     cout<<endl;
}
```

#OUTPUT:



```
student@student:~$ g++ -o openFact openFact.cpp -pg
student@student:~$ ./openFact
Dot Product is: 7138    1155    3255    7912    1029    1674    5310    1638    1
040     2592    748     1943    2460    1426    2345    58      1276    4623    5
208     462     2117    399     3108    2352    1050    338     7280    4088    4
340     7776    125     2268    180     1334    741     2280    3690    938     2
176     2150    696     5928    7392    153     5346    1920    5168    468     2
236     3666    6650    2652    67      194     1564    2912    80      3526    5
785     836     1160    527     6887    6075    243     3752    5141    5590    4
98      456     1988    928     57      4760    120     1960    2208    810     2
346     1155    6952    1792    2050    0       2176    336     4872    3913    1
755     2124    1632    1036    525     1554    5510    1073    3255    504     4
73      812     304     2709    494     240     72      2464    1173    1632    1
032     5810    8910    1800    3960    195     4644    5658    2688    679     2
20      528     616     4257    3128    880     110     5       1830    390     7
20      1144    1430    128     4756    888     1488    0       5148    3950    4
828     2263    2430    3102    3780    8019    9504    4307    884     8550    1
716     3360    7560    3192    252     2520    1422    1044    3456    531     3
60      3654    6       936     1155    1881    84      429     2680    140     1
350     4872    480     1518    7776    2520    6624    3600    2125    2178    1
680     1274    8820    2160    729     684     1760    376     5451    5548    2
750     2520    6636    465     1407    52      3294    1534    88      12      1
764     2856    2492    576     5684    288     2544    99      1584    4860    3
082     1972    0       8536    4410    99      6111    4876    2150    4992    6
600     1653    2160    294     240     756     2400    112     9118    4257    7
8       84      0       1786    3009    1190    3588    405     116     3136    2
465     1505    0       2698    4361    5896    8740    1892    2610    3280    2
829     832     2562    1020    1403    729     2250    6432    2618    2340    1
368     952     290     1032    0       2444    832     2262    4140    3290    4
080     2883    1539    1032    770     1080    948     690     6586    2255    6
```



```
696     5580    2015    3198    105     2695    3220    4015    153     1978    1
300     640     1746    4118    5110    5170    208     873     1975    60      2
72      4941    6873    2640    3108    295     2115    4875    3102    1408    5
81      6035    2967    6555    0       2835    380     2380    4592    1292    1
826     1050    4221    2867    8448    4290    23      6138    1936    805     6
12      2320    936     1288    1530    1518    612     1785    3672    1029    3
081     1736    1680    84      144     2522    3060    3686    252     0       4
347     1826    5694    57      480     360     396     6083    1508    7134    1
560     935     3600    3720    5115    120     888     4968    630     783     7
055     1836    7560    884     2016    140     5520    279     1729    1472    3
724     5402    4875    3040    7200    4851    2325    833     210     765     5
766     2175    444     324     177     3869    4900    6083    900     1425    2
200     770     4484    4085    5920    4664    850     3969    4653    48      1
067     3416    3360    418     282     2891    3450    2106    2146    1210    1
62      1376    767     2820    4118    4347    1420    1312    4234    480     7
189     567     2040    2460    2040    6432    520     2574    9702    351     1
38      2759    325     4814    7740    697     3969    4059    385     3360    3
007     3906    322     6512    324     2772    5427    444     1024    2982    2
590     4840    3276    3528    168     1980    2345    5551    98      221     8
1       690     400     3869    75      1896    3264    7569    352     354     2
232     5278    4686    510     1300    560     1378    1924    4653    4424    4
5       5742    4958    115     1054    2580    1920    450     704     3034
student@student:~$ gprof openFact > gprofile.txt
student@student:~$ ls
apache-tomcat-8.5.29  Documents    MNIST           Pictures   workspace
bin                   Downloads    Music           Project
CL1Practice           gmon.out     MyServer.java   Public
CODES                 gprofile.txt openFact        Templates
Desktop               Home         openFact.cpp    Videos
student@student:~$ █
```

File: gprofile.txt

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total           
 time   seconds   seconds    calls  ms/call  ms/call  name    
100.83      0.01     0.01        1    10.08    10.08  initialize()
  0.00      0.01     0.00        1     0.00     0.00  _GLOBAL__sub_I_a
  0.00      0.01     0.00        1     0.00     0.00  
__static_initialization_and_destruction_0(int, int)
  0.00      0.01     0.00        1     0.00     0.00  display()
  0.00      0.01     0.00        1     0.00     0.00  Multiply()

 %         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.

Copyright (C) 2012-2015 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.

               Call graph (explanation follows)


granularity: each sample hit covers 2 byte(s) for 99.18% of 0.01 seconds

index % time    self  children    called     name
                                                 <spontaneous>
[1]    100.0    0.00    0.01                 main [1]
                0.01    0.00       1/1           initialize() [2]
                0.00    0.00       1/1           display() [11]
-----------------------------------------------
```

```
                0.01    0.00       1/1           main [1]
[2]    100.0    0.01    0.00       1         initialize() [2]
-----------------------------------------------
                0.00    0.00       1/1           __libc_csu_init [18]
[9]      0.0    0.00    0.00       1         _GLOBAL__sub_I_a [9]
                0.00    0.00       1/1
__static_initialization_and_destruction_0(int, int) [10]
-----------------------------------------------
                0.00    0.00       1/1           _GLOBAL__sub_I_a [9]
[10]     0.0    0.00    0.00       1
__static_initialization_and_destruction_0(int, int) [10]
-----------------------------------------------
                0.00    0.00       1/1           main [1]
[11]     0.0    0.00    0.00       1         display() [11]
                0.00    0.00       1/1           Multiply() [12]
-----------------------------------------------
                0.00    0.00       1/1           display() [11]
[12]     0.0    0.00    0.00       1         Multiply() [12]
-----------------------------------------------

 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
          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.

Copyright (C) 2012-2015 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

   [9] _GLOBAL__sub_I_a       [10]
__static_initialization_and_destruction_0(int, int) [12] Multiply()
   [2] initialize()           [11] display()
```