

# DATA STRUCTURES & ALGORITHMS 1

BATCH – B

[SUNDAY MARCH 24, 2019: 2:30 PM – 5:30 PM]

LAB ASSIGNMENT – 7

CODE:assign07

NOTES:

1. Please carefully read all assignments and there is **no choice**.
  2. **Use the template for this assignment**
  3. Each problem in this assignment has to be answered in the **same c file**.
  4. Create a .c file following the **file name convention**:
    - a. If your roll number is 'abc' and assignment code is 'assignXX'. Then use the following file name convention as follows: 'abc-assignXX.c'
    - b. For example, if the roll number is 92 and assignment code is assign01, then the file name should be 092-assign01.c
    - c. Strictly follow the file name convention. When you are ready, submit the solution via google classroom.
  5. **Follow variable and function naming conventions**
    - a. except for variables in for-loop, none of the other variables should be a single character.
    - b. The variable names and function names should indicate what they are storing/computing. For this assignment, we have given you some of the variable names and function names to use. They are highlighted as **function\_name** or **variable\_name**
    - c. All global variable should start with '**g\_**'
  6. Indentation improves readability. Please pick an indentation style and **indent your code** appropriately.
  7. Follow constants and type naming
    - a. All **constants** should be defined using **IFDEF and DEFINE**
    - b. All **structures** should have a **TYPDEF** to a simpler name
  8. When in doubt about naming or style conventions, consult the following link:  
<https://users.ece.cmu.edu/~eno/coding/CCodingStandard.html>
-

```

#include <iostream>
#include <time.h>
using namespace std;

timespec diff(timespec start, timespec end);

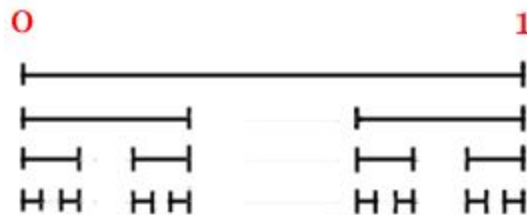
int main()
{
    timespec time1, time2;
    int temp;
    clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &time1);
    for (int i = 0; i < 242000000; i++)
        temp+=temp;
    clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &time2);
    cout<<diff(time1,time2).tv_sec<<": "<<diff(time1,time2).tv_nsec<<endl;
    return 0;
}

timespec diff(timespec start, timespec end)
{
    timespec temp;
    if ((end.tv_nsec-start.tv_nsec)<0) {
        temp.tv_sec = end.tv_sec-start.tv_sec-1;
        temp.tv_nsec = 1000000000+end.tv_nsec-start.tv_nsec;
    } else {
        temp.tv_sec = end.tv_sec-start.tv_sec;
        temp.tv_nsec = end.tv_nsec-start.tv_nsec;
    }
    return temp;
}

```

PROBLEMS [Total Marks: 20]:

The above code snippet shows an example of how you can profile your code using time.h. Even though, the above code is in C++, it's the same process for C. Feel free to use an alternative approach.



The structure for cantor set would contain { start, end, next\_line }. Where start & end are the starting and ending point of the line segment, next\_line is the next line segment.

1. You should implement a way to create a cantor set and store it in a linked list. The level 'L' is given as the input. The image above shows line segments upto level 3 (the starting segment (0,1) is level 0. [6 Marks]

2. Please profile your code. You can use the technique shown in the snippet above (or) some other technique for profiling the code [6 Marks]
3. Improve your code, so you can either pass the level 'L' (or) the 'max\_line\_segments' as the input. (For example, if the max\_line\_segment = 3, then you will stop at level 1, where you will have 2 line segment) [2 marks]
4. Please improve your code to take 'L' (or) 'max\_line\_segments' as the command line argument[ 4 Marks]
5. Rewrite cantor set generation using recursion [2 Marks]