BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI,
K. K. BIRLA Goa Campus,

I SEMESTER 2014 – 2015

Advanced Operating Systems (CS G623)

Assignment 3

Due date 30/09/2014 (12:00 Midnight)

Instructions: Please mail your completed assignment to biju@goa.bits-pilani.ac.in
The programming assignments will be graded according to the following criteria

- *Completeness; does your program implement the whole assignment?*
- *Correctness; does your program provide the right output?*
- *Efficiency; have you chosen appropriate algorithms and data structures for the problem?*
- *Programming style (including documentation and program organization); is the program well designed and easy to understand?*
- *Individual Viva.*

Submit your assignment as a tar.gz file.
DO NOT FORGET to submit a README file (text only) with following contents.

General README instructions

1. *Your name. If you interacted significantly with others, indicate this as well.*
2. *A list of all files in the directory and a short description of each.*
3. *HOW TO COMPILE your program (You need to submit MAKEFILE for each question).*
4. *HOW TO USE (execute) your program.*
5. *A description of the structure of your program.*
6. *In case you have not completed the assignment, you should mention in significant detail:*
    a. *What you have and have not done,*
    b. *Why you did not manage to complete your assignment (e.g., greatest difficulties)*
7. *This will allow us to give you partial credit for the things you have completed.*
8. *Document any bugs in your program that you know of. Run-time errors will cost you less penalty if you document them and you show that you know their cause. Also describe what you would have done to correct them, if you had more time to work on your project.*

NB: Late submission will not be entertained.

QUESTION 1: ***From Spiderman to the Avengers!!!***

Nick Fury has put a team together to get a hold of the stolen Tesseract. Using clues obtained by Gamma radiations from Tesseract, intelligent Bruce Banner and Tony Stark have provided Fury with a sequence of clues about the location. Each digit in the clue enables each member to search the location efficiently. Using the sequence Nick Fury wants an immediate result on the location of the Tesseract. Find out whether the clues sequence can successfully identify the location or not. For faster and efficient interpretation of the clues sequence, the following plan has been devised.

Input: (Refer figure at the end)

- NxN matrix with entries from 0 to 9 denoting available locations where (0,0) entry is location of the Avengers and (N-1, N-1) is location of Tesseract.
- And array of clues with each entry between 0 to 9.

Write a program to determine whether the given array sequence finds the final location or not. The sequence is said to be valid iff there exists at least one path from (0, 0) to (N-1, N-1) that covers all the digits in the given sequence in the *same order* as they appear.

Meaning,

(0, 0) entry in matrix entry should match with 0^{th} digit of the sequence.

(0,1) **or** (1,0) **or** (1,1) entry in the matrix matches with 1^{st} digit and so on till (N-1, N-1) entry matches with the last digit of the sequence.

You are expected to create threads for each entry in the matrix to ensure parallel scanning.

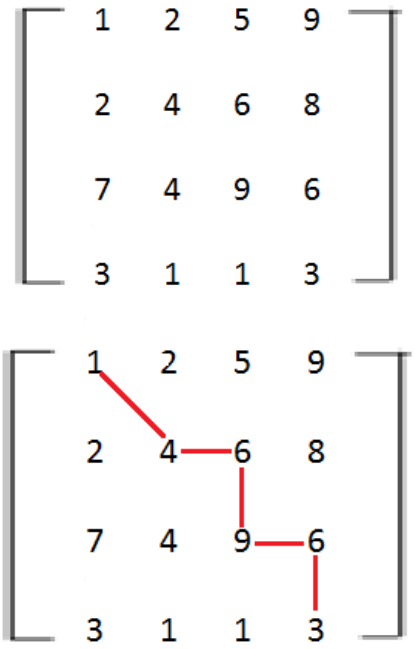
Procedure:

- The main thread will read the entries in the matrix and will check for the first entry of the matrix with the first entry of the array. If it matches, three threads will be created.
  - First one will check the entry to the right of the current entry.
  - Second one will check the entry below the current entry.
  - Third will check the entry diagonal to the current entry.
- If the entry doesn't match, the thread will terminate and returns termination status.
- For, each entry of the matrix, the same above step will take place. If the entry is valid at (i,j), we will have three threads that will verify the (i+1,j), (i,j+1) and (i+1,j+1) entries if at all these entries exist. For boundary cases, i.e. the last row and the last column the cases should be handled properly and no extra threads should be created.
- The threads should keep on executing till either a valid path is found or an invalid entry is encountered. Corresponding to each condition, respective values should be returned while exiting (1 for a success and 0 for a failure). The parent thread should use this status to verify the existence of the path.

- If the path exists i.e. (N-1, N-1) entry matched with the last element of the sequence, then ***all*** such paths should be printed.

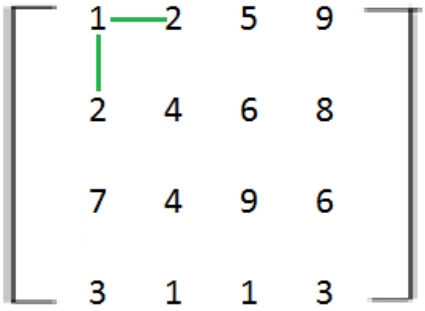
Output: The output must be printed in **reverse order** where in each thread participating in the actual path prints its (i,j) pair.
Please follow the examples given below.

Example 1:
Sequence: **1 4 6 9 6 3 (unique path)**



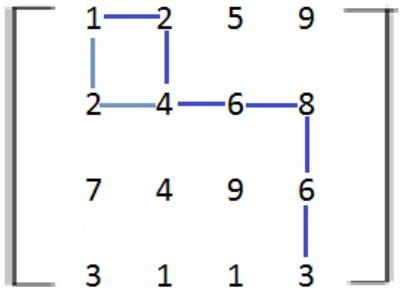
Output : (3,3)→(2,3)→(2,2)→(1,2)→(1,1)→(0,0)

Example 2:
Sequence: **1 2 3 4 5 6 7 (no path)**



Output: No path exists

Example 3:



Sequence: **1 2 4 6 8 6 3**
Output :

(3,3)→(2,3)→(1,3)→(1,2)→(1,1)→(0,1)→(0,0)
(3,3)→(2,3)→(1,3)→(1,2)→(1,1)→(1,0)→(0,0)

Input File Format:
<Number of elements in sequence array = M>
<Sequence array>
<Size of square matrix = N>
<Matrix>

QUESTION 2: ***Multiplication of Complex Numbers***

Write a program to multiply N complex numbers using threads.

Input File Format:
<No. of complex numbers = N>
<Complex No. 0>
<Complex No. 1>
:
:
<Complex No. N-1>

The complex number will be of the form **"a+ib"** where a and b are integers.
For e.g.
1 + i2 (a = 1, b = 2)
1 + -1i (a = 1, b = -1)
-1 + i4 (a = -1, b = 4)

Procedure:

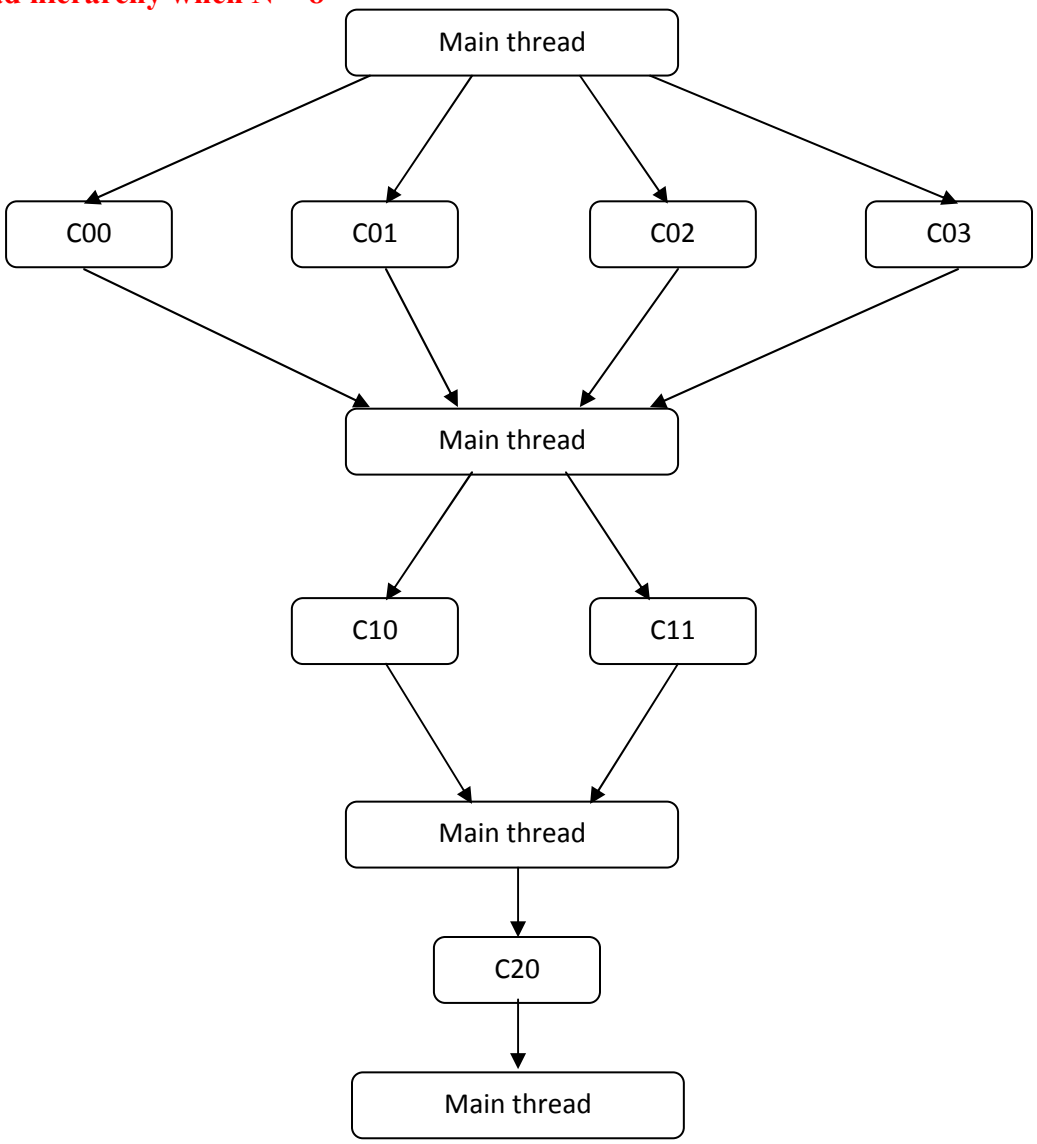
Each complex number is stored as a structure with a real and an imaginary integer member in it. The main thread should read N complex numbers from the file and store it as D[N] an array of structure in global data space.
The program should do the following

- Create [floor (N/2)] number of threads from the main thread
  - Each thread will compute product of $(2*i)^{th}$ input and $(2*i+1)^{th}$ input where i is the thread number [0 to [floor (N/2)]-1]. The product is stored in temporary output array in global space.

- If N is an even number, the number of threads created for computation is equal to N/2 and the temporary output array elements will be equal to N/2.
- If N is an odd number, the number of threads created for computation is equal to [floor (N/2)] and the temporary output array elements will be equal to [ceil (N/2)]. This is because the last input is transferred directly as output for the next processing.

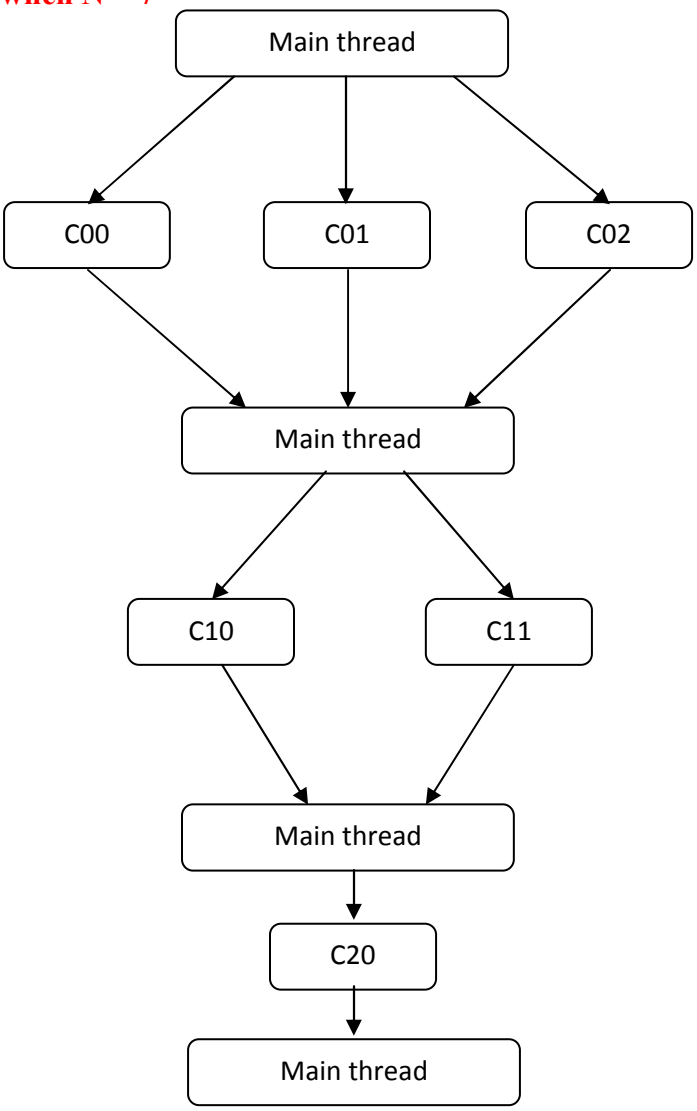
- After these [floor (N/2)] threads exit, the main process will print the intermediate results and will use this as the next input for the processing.
- This process will continue till we get the final result, i.e., the number of elements in inter mediate output is exactly 1.
- Print the final result from the main thread.

Thread hierarchy when N = 8



N = 8 will have 8 complex numbers (CN) as input
C00 computes the product of CN 0 and CN 1, C01 computes the product of CN 2 and CN 3
C02 computes the product of CN 4 and CN 5, C03 computes the product of CN 6 and CN 7
C10 computes the product of CN 0, CN 1, CN 2 and CN 3
C11 computes the product of CN 4, CN 5, CN 6 and CN 7
C20 computes the product of CN 0, CN 1, CN 2, CN 3, CN 4, CN 5, CN 6 and CN 7

Thread hierarchy when N = 7



N = 7 will have 7 complex numbers (CN) as input
C00 computes the product of CN 0 and CN 1, C01 computes the product of CN 2 and CN 3
C02 computes the product of CN 4 and CN 5
C10 computes the product of CN 0, CN 1, CN 2 and CN 3
C11 computes the product of CN 4, CN 5 and CN 6
C20 computes the product of CN 0, CN 1, CN 2, CN 3, CN 4, CN 5 and CN 6

Question 3: ***Multiplication of two N digit Numbers***

Write a program in C to demonstrate multiplication of two N digit numbers. The computation for each digit in the result must be performed by one thread. The final generation of the result (along with the carry forwarding) will be taken care by the main thread.

Example:

```
                    1      2      3      X
                    4      3      1
========================================
                    1   |  2   |  3
              3  |  6   |  9
        4  |  8  |  12
================================
        4  | 11 |  19   |  11  |  3      (returned by threads t1 to t5)
================================
        5  |  3 |  0    |  1   |  3      (computed by main)
```