This assignment is to be done individually. Here are some rules regarding the assignments in this course:

- You may consult with other students about:
  - Use of compiler, debugger or general C programming issues.
  - UNIX shell user-commands.

- You can't get help from other students for fixing errors in your program or for solving the assignment.

- Rules regarding plagiarism are in the Student Handbook

General Instructions:

For this assignment, your accounts have been created on the LUMS cluster. Your login is aRoll_Number e.g. a05030037. These accounts have been created for work related to this course only and will be deleted one week after the final exam. Do not place any personal files or material related to other courses here, since they will be deleted without notice and your account will be deactivated.

Important:
Make a directory called cs524a1 where you will place your code for this assignment. Turn off access permissions to this directory so that other users cannot read your code. i.e. do the following:
mkdir cs524a1
chmod go-rwx cs524a1

Failure to turn off the access permissions will be seen as an attempt to help others cheat and will result in zero marks in your assignment.

For examples of computing elapsed time, two files: walltime.h and sample.c are provided with this assignment.

The LUMS cluster has 8 computing nodes with Intel dual-Xeon processors (16 CPUs in all). The nodes are interconnected by a switched gigabit network. The cluster front-end is accessible from outside has an IP address of 203.128.0.145. The rest of the seven compute nodes are only accessible through the front-end.
Different students will be using different nodes for doing this assignment. To calculate your node number, add up all the digits in your roll number and divide by 7 (e.g. 2005-03-0007 adds to 17 and dividing by 7 gives a remainder of 3). The remainder will determine which node you will be running your experiments.

Remainder: 0  Node: c0-0
Remainder: 1  Node: c0-1
Remainder: 2  Node: c0-2
Remainder: 3  Node: c0-3
Remainder: 4  Node: c0-4
Remainder: 5  Node: c0-5
Remainder: 6  Node: c0-6

You will first connect through ssh to the frontend (i.e. 203.128.0.145) and do your coding and compiling there. For running your executable on the compute node, you need to connect to it by typing:

```
ssh NodeNo  
```

(e.g. a person with roll no 2005-03-0007 will type `ssh c0-3`)

It is easier if you have a second console window open that you will use to connect to your compute node. After connecting to the compute node go to the directory where your executable resides and run it.
Interconnection networks are used to connect components in shared-memory and distributed memory parallel systems. They range from simple buses, 2-D and 3-D meshes to complex hypercube network topologies. When interconnects are used to implement shared memory systems, they are designed to operate at memory system speeds and low latencies. Interconnects, as networks between computers, on the other hand, have higher latencies and lower bandwidths.

For the following interconnects, you have to write one paragraph each, listing their key features and the kind of parallel systems they are used to build. Also specify the number of systems in the current top500 list that are using each of the following interconnect technology (Refer to www.top500.org).

- Infiniband
- Myrinet
- Scalable Coherent Interconnect (SCI)

Clearly identify the sources used for doing this part. The write-up must be in your own words.

In this part you will write code to estimate the following for a computer.

i) Memory bandwidth
ii) Cache size.

(Hint: In order to estimate memory bandwidth, you do not want data reuse, but for estimating cache size you want to see the effect of reuse. Try accessing increasing amounts of data until you see a sharp decline in reuse. Plot the data sizes used against the performance in MFLOPS).

What you will submit:
- Your code (it should be well commented).
- Your strategy/logic in solving this problem and the conclusions drawn from it.
- Your measured results for parts (i) and (ii) and plot for (ii).

In this part you will experiment with the use of blocking for decreasing the number of cache misses in nested loops. Consider the standard O(n^3) matrix multiplication algorithm:

```c
for (i=0; i<n; i++) {
    for (j=0; j<n; j++) {
        for (k=0; k<n; k++) {
            c[i][j]=c[i][j] + a[i][k]*b[k][j];
        }
    }
}
```
Take the matrix size $n=2048$. Compare the difference in performance (in terms of time taken and MFLOPS) for unblocked and blocked matrix multiplication. You will experiment with different block sizes ($S$).

**Note:** In the example discussed in class, we used $\text{min()}$ function in the inner three loops. You will change the code so that $\text{min()}$ function is not used.

What you will submit:
- Your code (it should be well commented).
- Your strategy/logic in solving this problem and the conclusions drawn from it.
- Your measured results. Plot the block sizes ($S$) used against the performance in MFLOPS.

---

## Part IV

Marks: 25

In this part you will experiment with the use of loop unrolling. Consider the following code.

```c
for (i=0; i<n; i++) {
    for (j=0; j<n; j++) {
        for (k=0; k<n; k++) {
            c[i][j] = c[i][j] + a[i][k] * b[k][j];
        }
    }
}
```

Take $n=2048$.

1. Determine the running time and MFLOPS of the above code fragment both with and without compiler options for loop unrolling. Does using the compiler option improve performance? (Consult the compiler documentation for the appropriate option to use).

2. For the code given above, manually unroll the $i$-loop by 2 and unroll the $j$-loop by 4 and determine the improvement in performance (in terms of time taken and MFLOPS) compared to the original.

What you will submit:
- Your code (it should be well commented).
- Your strategy/logic in solving this problem and the conclusions drawn from it.
- Your measured results.

---

## Part V

Marks: 15

Do problems 2.2, 2.3 and 2.4 of your textbook. Pages 76-77.
**Submission Instructions:**

- You have to submit your source code along with the makefile (if applicable).
- Make sure you have your name and roll number at the beginning of each file submitted.
- Your code must be properly commented. There are marks for style and quality of coding.
- Instructions about how and where to submit your assignment will be posted on the course website.
- Vivas will be scheduled for each student while grading your assignments. Vivas are mandatory and information about viva schedule will be made available later through the website.
- Your answers should be clearly typed and submitted as a text/doc file.