PROBLEM 1

Question 1.

$/*\ A\ is\ the\ input\ matrix,\ \ I\ is\ \ the\ inversed\ matrix\ */$

$void\ inverse\ (I, A, i, j, k, l, N, Base)$

$\quad if(n\ \le Base)$

$\qquad invoke\ loop\_serial\_inverse(A, IA, N)\ \ to\ get\ inverse\ of\ the\ matrix$

$\quad else$

$\qquad spawn\ inverse\ \left(I, A, \frac{n}{2}, Base\right) \qquad /*\ \ computer\ A_1^{-1}\ by\ recursive\ call$

$\qquad spawn\ inverse\ \left(I, A, \frac{n}{2}, Base\right) \qquad /*\ \ computer\ A_3^{-1}\ by\ recursive\ call$

$\qquad sync$

$\qquad spawn\ \ Multi(\ W, -\ A_3^{-1} A_2\ ) \qquad /*\ Copy\ value\ to\ W$

$\qquad spawn\ \ Multi(\ D, W\ A_1^{-1}) \qquad /*\ Copy\ value\ to\ D$

$\qquad spawn\ \ CopyMatrix(\ A_2, D) \qquad /*\ Copy\ value\ to\ A_2$

$\qquad sync$

$\quad return$

$***\ \ The\ naive\ iterative\ method\ that\ implements\ matrix\ inversion\ using\ forward\ substitution\ for\ a\ lower\ triangular\ matrix$

Question 2.

For the analysis of the work and the critical path, we neglect the subroutines for simplicity. Based on the algorithm pseduo code, Inverse function and Multiplcation functions

The work $I_1(n)$ *satisifers*:

$$
\begin{aligned}
I_1(n) &= 2I_1\left(\frac{n}{2}\right) + 2M_1\left(\frac{n}{2}\right) \\
&= 2I_1\left(\frac{n}{2}\right) + \Theta(n^3) \\
&= \Theta(n^3) \qquad /* \text{ according to the Master's Theroem}
\end{aligned}
$$

The critical path $I_\infty(n)$ *satisifers*:

$$
\begin{aligned}
I_\infty(n) &= 2I_\infty(n/2) + 2M_\infty(n/2) \\
&= 2I_\infty(n/2) + 2\Theta(log2(n/2)) \\
&= \Theta(n)
\end{aligned}
$$

Question 3.

Code is attached q1.cpp

Markfile is attached as well

- *Make clean*
- *Make all*
- *Make test*

Due to precision error, some test case may not get identical matrix. But I built some unit testing to test each function in the code. Using the three matrix provided by assigment, all test cases are passed.

Question 4.

$My\ Experimentally\ Data\ to\ find\ optimal\ B\ is$

$$\mathrm{B} = \{32, 64, 128, 256, 512\}$$

$$\mathrm{N} = \{4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$$

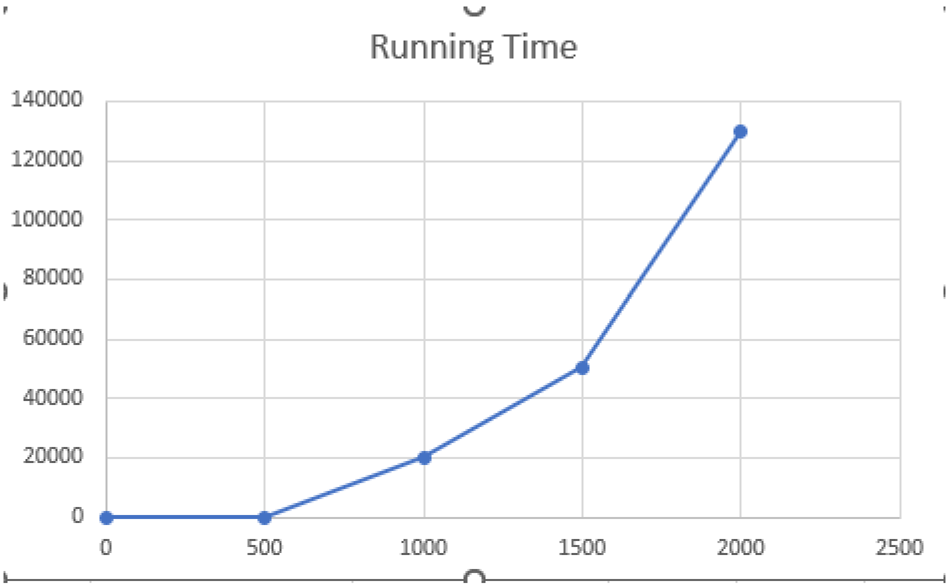
Serial implementation suggests B value of 256 is best shown by the different resulting numbers achieved from the optimal B value.

	Serial Implementation				
	B=32	B=64	B=128	B = 256	B= 512
n= 4	12	10	8	7	6
n=8	15	13	11	10	9
n=16	21	18	16	14	13
n=32	30	27	24	22	21
n=64	45	41	37	34	32
n=128	79	71	64	59	56
n=256	152	139	124	112	106
n=512	302	275	247	223	211

	Parallel Implementation				
	B=32	B=64	B=128	B = 256	B= 512
n= 4	1	3	6	12	24
n=8	1	2	5	10	20
n=16	10	18	16	14	13
n=32	30	27	24	22	21
n=64	45	41	37	34	32
n=128	25	30	28	40	36
n=256	152	139	124	112	106
n=512	283	210	247	223	211

Question 5.

$My\ processor\ is\ 12th\ Gen\ Inte(R)Core(TM)i7 - 1255U. Hyper - threading\ is\ available\ on\ my\ process\ and\ it\ is\ also\ enabled$



(X -Axis Size, Y-Axis Runing Time) (Blue is serial in ms and parallel is a straight line through 0)

Question 6.

$To\ analyze\ the\ Cache\ Complexity\ of\ the\ serial\ elision\ of\ the\ algorithm,$
$we\ should\ consider\ the\ serial\ version\ of\ multipliation. For\ square\ matrices$
$of\ order\ n\ we\ have$:

$$Q(n, Z, L) = \begin{cases} O\left(n + \frac{n^3}{L}\right) & if\ \ 3L \le Z < n^2 \\ O\left(1 + \frac{n^2}{L}\right) & if\ 3n^2 \le Z \end{cases}$$

Reference: Lecture Note - Cache Memories, Cache Complexity

PROBBLEM 2

Question 1.

$in: A\ set\ of\ n\ points$
$out: The\ closest\ pair\ of\ points\ \{p_i, p_j\}$

$vector < tuple < int, int \gg csp(vector < tuple < int, int \gg points)$
{
$//$
$sort\ x\ and\ find\ the\ middle\ position\ to\ split\ the\ points\ into\ L\ and\ R\ group$

sort_x()

$int\ pivot = number\ of\ points/\ 2$

$vector < tuple < int, int \gg l_{group} = get\_left\_group\_points()$

$vector < tuple < int, int \gg r_{group} = get\_right\_group\_points()$

$//\ apply\ divide\_and\_conquer\ approach\ to\ get\ the\ shortest\ distance$

$d_L = csp(left)$ $//\ get\ left\ distance$

$d_R = csp(Right)$ $//\ get\ right\ distance$

$find\_Points\_p_l\_q_r()$ $//\ get\ middle\ (cross)\ points$

$d_m = Points(p_l, q_r)$ $//\ get\ right\ distance$

sort_y()

$d_{min} = minumn(d_L, d_{R,} d_M)$ $//\ get\ the\ shortest\ distance$

}

$There\ are\ 2\ recurrsive\ calls\ in\ this\ function\ since\ the\ divide\_and\_conquer$ $algorithm\ is\ applied.\ Therefore\ it\ took\ 2W(^n/_2)$

$Sorting\ functions\ are\ used\ in\ the\ solution\ and\ which\ will\ take$ Θ(nlog(n). Therefore, the total work is given by $W(n) = 2W(n/2) + O(nlog(n))$.

Deduce that $W(n) \in O(nlog^2(n))$

Question 2.

$in: A\ set\ of\ n\ points$
$out: The\ closest\ pair\ of\ points\ \{p_i, p_j\}$

$vector < tuple < int, int \gg csp(vector < tuple < int, int \gg points)$
{
$//\ sort\ x\ and\ find\ the\ middle\ position\ to\ split\ the\ points\ into\ L\ and\ R\ group$

sort_x()

$int\ pivot = number\ of\ points / 2$

$vector < tuple < int, int \gg l_{group} = get\_left\_group\_points()$

$vector < tuple < int, int \gg r_{group} = get\_right\_group\_points()$

$//\ apply\ divide\_and\_conquer\ approach\ to\ get\ the\ shortest\ distance$

cilk_spawn $d_L = csp(left)$ $//\ get\ left\ distance$

$d_R = csp(Right)$ $//\ get\ right\ distance$

cilk_sync

$find\_Points\_p_l\_q_r()$ $//\ get\ middle\ (cross)\ points$

$d_m = Points(p_l, q_r)$ $//\ get\ right\ distance$

sort_y()

$d_{min} = minumn(d_L, d_{R,} d_M)$ $//\ get\ the\ shortest\ distance$

}

$Apply\ multi\_threaded\ version\ of\ algorithm\ by\ adding\ cilk\_spawn\ and$

$cilk\_Sync$

$The\ Work\ W\ and\ Span\ S$:

$$W(n) = 2W(n/2) + O\big(nlog(n)\big) = O(nlog^2(n))$$

$$S(n) = 2S\left(\frac{n}{2}\right) + O\big(nlog(n)\big) = O\big(nlog(n)\big)$$

$$parallelism = \frac{W(n)}{S(n)} = \frac{O\left(nlog^2(n)\right)}{O\left(nlog(n)\right)} = O\big(log(n)\big)$$

Question 3.

$We\ can\ imporoved\ algorithm\ on\ 2\ sorting\ functions. If\ we\ implement\ it\ as\ a\ Parallel\ Merge\ Sort, it\ will\ definitly\ get\ a\ better\ paralleim.$

$The\ work\ W\ has\ no\ change, but\ the\ span\ S\ will\ bechange\ to\ O(log^3(n))$

$$parallelism = \frac{W(n)}{S(n)} = \frac{O\left(nlog^2(n)\right)}{O\left(log^3(n)\right)} = O(\frac{n}{logn})$$

Question 4.

Attached q2.cpp