Question 4:
The code for the problem is present in the attached github repository. We could note that we have 3 nested for loops and we can interchange them maintaining the precision of the solution. Following are the combinations of the loop and their performance

Note: *Time in Seconds*

I, J, K

N	128	256	512
Iteration 1	0.28224969199982297	2.073409698999967	16.996527972000877
Iteration 2	0.26032037300046795	2.0517706720002025	16.78452467199986
Iteration 3	0.26722570800029644	1.99435360499956	17.841820249000193
Median	**0.26722570800029644**	**2.0517706720002025**	**16.996527972000877**

I, K, J

N	128	256	512
Iteration 1	0.25056748799943307	1.8869987930002026	13.756732945000294
Iteration 2	0.2495645169992713	1.8789108379996833	14.168155080999895
Iteration 3	0.2748677259996839	1.9172490860000835	15.733258565000142
Median	**0.25056748799943307**	**1.8869987930002026**	**14.168155080999895**

K, I, J

N	128	256	512
Iteration 1	0.26016236700070294	1.9112885190006637	13.158623727999839
Iteration 2	0.2540488489994459	1.9509402309995494	14.424502025999573
Iteration 3	0.2537215259999357	1.9074726670005475	14.70270794499993
Median	**0.2540488489994459**	**1.9112885190006637**	**14.424502025999573**

K, J, I

N	128	256	512
Iteration 1	0.26719552600025054	1.995343726000101	17.234896037999533
Iteration 2	0.2630918090007981	2.048039387999779	17.18846589900022
Iteration 3	0.26402612299898465	2.1766023209993364	16.882021737000287
Median	**0.26402612299898465**	**2.048039387999779**	**17.18846589900022**

J, K, I

N	128	256	512
Iteration 1	0.28957370899843227	2.091274442998838	19.85446316099842
Iteration 2	0.2797036539996043	2.0989001909983926	19.695707141998355
Iteration 3	0.2809136839987332	2.059013183999923	19.81430137800089
Median	**0.2809136839987332**	**2.091274442998838**	**19.81430137800089**

J, I, K

N	128	256	512
Iteration 1	0.307991289999336	2.116191377999712	17.332238044000405
Iteration 2	0.27211802800047735	1.9963988079998671	15.595265014999313
Iteration 3	0.27281849100108957	2.0243031270001666	15.420025614999759
Median	**0.27281849100108957**	**2.0243031270001666**	**15.595265014999313**

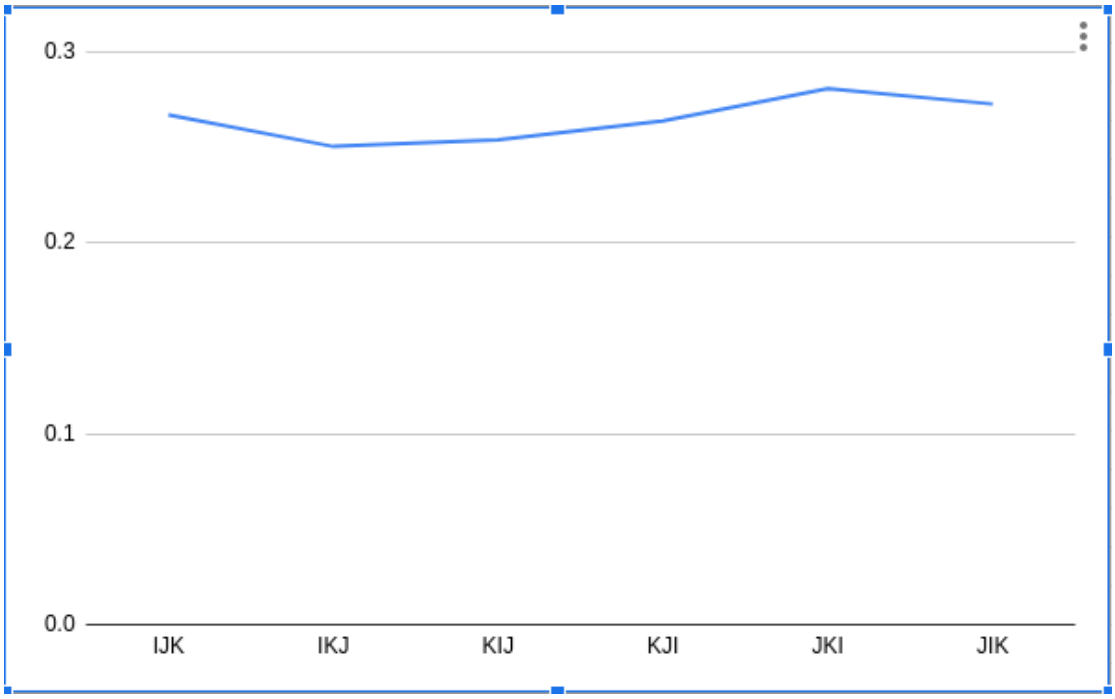
SUMMARY:

The summary of the above experiment is shown in the following table:

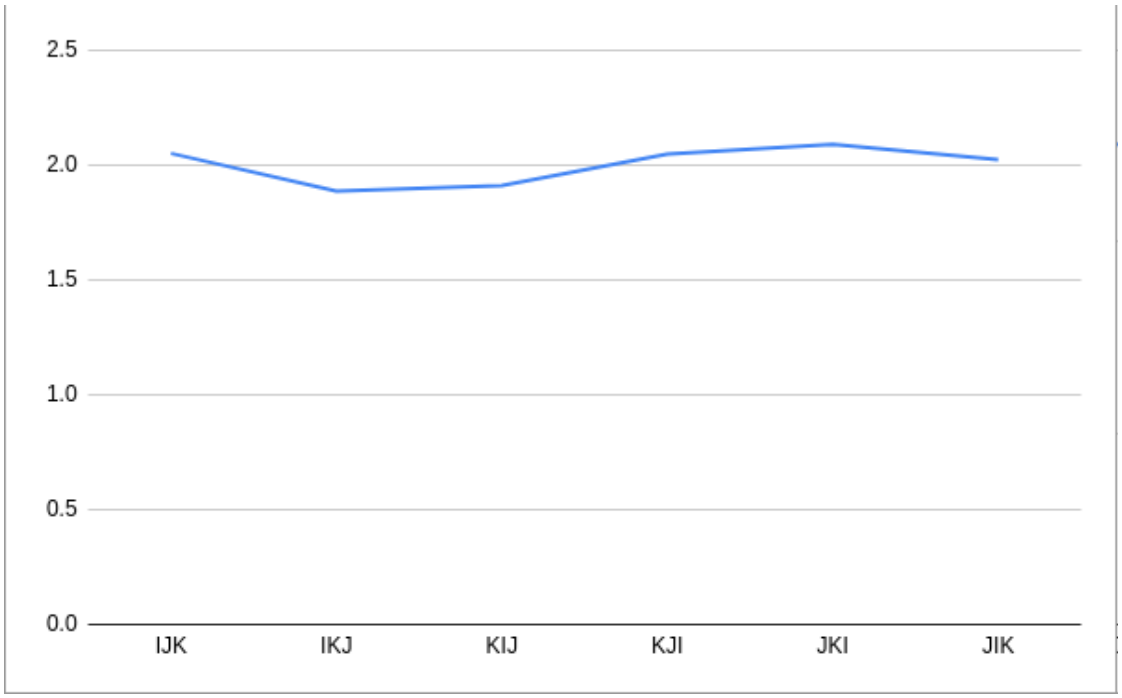
	128	256	512
IJK	0.267225708	2.051770672	16.99652797
IKJ	0.250567488	1.886998793	14.16815508
KIJ	0.254048849	1.911288519	14.42450203
KJI	0.264026123	2.048039388	17.1884659
JKI	0.280913684	2.091274443	19.81430138
JIK	0.272818491	2.024303127	15.59526501

Graphs(*showing variation of N with permutation of i, j, k***)**

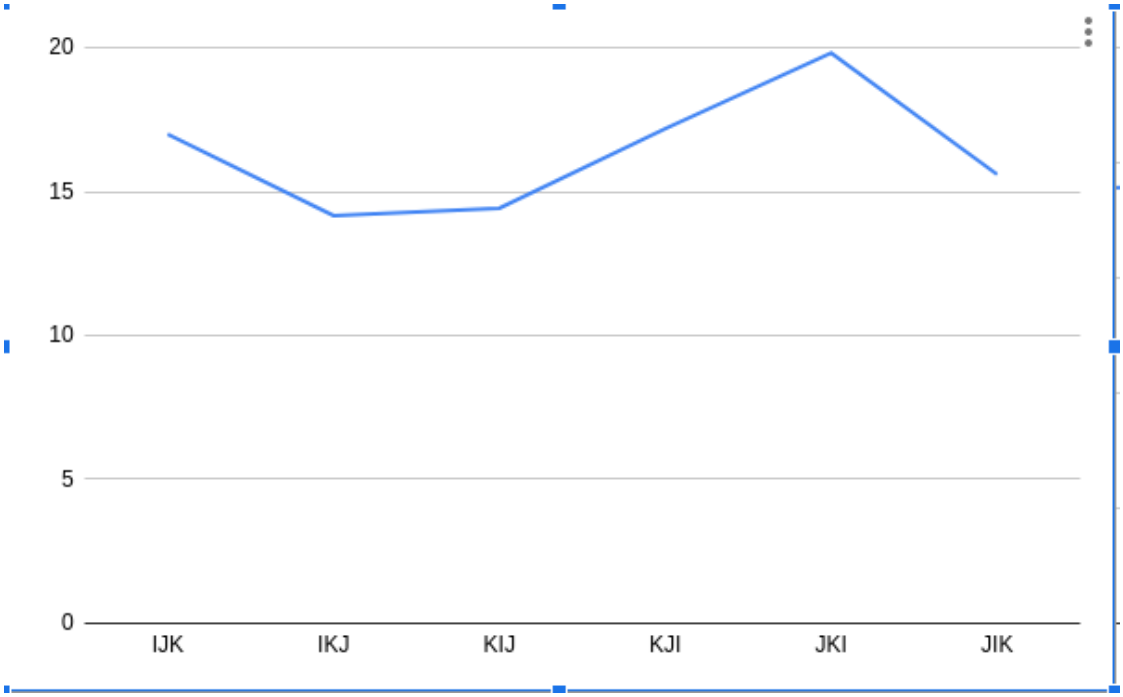
N = 128



N = 256



N = 512



SUMMARY

All the observations noted here have above graphs and experimental experience as basis

1. The overall **behavior** of the graph for all values of N remains **similar,** but we observe significant deviation for N = 512.
2. The code execution in python takes a lot longer than code execution in C++, which was expected.
3. We observe a steep slope for N = 512, which is absent in rest two.
4. **Minimum** time(best sequence) = **i, k, j**
5. **Maximum** time(worst sequence) = **j, k, i**
6. The performance of **k, i, j** and **i, k , j** are very close and in most cases the difference isn't significant. Unlike for C++, the difference of performance between **j, k, i** and **k, j, i,** isn't negligible.
7. Most of the performance is determined by the inner most loop:

$$Performance(as\ inner\ most\ loop): j > i > k$$