Q2. Loop interchange advantage:

* Interchange is transformation applied to nested loops in which order of the loops is changed.

* Ex: 1

- Switch the nesting order of loops in a perfect loop nest.
- Can increase parallelism, can improve cache locality.

```
      DO I = 1 , N
        DO J = 1, M
S:    A(I, J+1) = A(I, J) + B
      ENDDO
      END DO
```

```
      DO J = 1 , M
        DO I = 1 , N
S:    A(I, J+1) = A(I, J) + B
      END DO
      END DO
```

* Ex: 2:

- Interchange of NON-rectangular loops

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
    y[i] = y[i] + A[i][j] * x[j];

for (j = 0; j <= n; j++)
    for (i = j + 1; i <= n; i++)
    y[i] = y[i] + A[i][j] * x[j];
```

Ex:3
```
for (i=0; i<n; i++)
    for (j=0; j<n; j++){
        a[j][i+1] = 2.0 * a[j][i-1];
    }
```

```
for (j=0; j<n; j++)
    for (i=0; i<n; i++)
        a[j][i+1] = 2.0 * a[j][i-1];
    }
```

// Compiler optimization using O1, O2, O3, Os, Ofast etc :

- without any optimization option, the compiler's goal is to reduce the cost of compilation and to make debugging produce the expected results.

- O0 → No optimization; generates un-optimized code but the fastest compilation time.
- O1 → made code optimization; optimizes reasonably well but doesn't degrade compilation time significantly.
- O2: full optimization: generates highly optimized code and has the slowest compilation time.

- **-O3**: full optimization as in -O2; also uses more aggressive automatic inlining of subprograms within a unit and attempts to vectorize loops.
- **-Os**: optimize space usage of resulting program.

- Higher optimization levels perform more global transformations on the program and apply more expensive analysis