Problem Set 1

A)

a)

Pseudo code :-

```
PARTITION (r,L)

        cubesContainingR = { }

        for (l1 = 0 to L-1)
                for(l2 =0 to L-1)
                        for(l3 = 0 to L-1)
                                l = (l1,l2,l3)
                                if  r belongs to Cl
                                        add l to cubesContainingR
                        end for
                end for
        end for
        return cubesContainingR
end PARTITION

PARTITION-ALL(R,L)
        INPUT  – R: set of K points in C
                     L:  positive integer
        Rl[L][L][L][K] <- {0}
        for (r in R)
                cubesContaining_r <- PARTITION(r,L)
                for (l in cubesContaining_r)
                        Rl[l.x][l.y][l.z][indexof(r)] <- 1
                end for
        end for
        for (l in Cl)
                write l to file
                for(r in R)
                        if (Rl[l.x][l.y][l.z][indexof(r)] == 1)
                                write r to file
                end for
        end for
end PARTITION-ALL
```

Algorithm Analysis :- For each of l_1,l_2 and l_3 which ranges from **0 to L-1,** check whether r belongs to C_l and if so then save it. So there exists a total of $\mathbf{L^3}$

combinations for each l_1, l_2, l_3 .so there exists **L^3** opeatations and hence the complexity of the algorithm is **$O(L^3)$.**

b)

<u>Pseudo Code:-</u>

```
BUILD-MATRIX (L, l, P, Q)
        INPUT – L: positive integer
                 l: (l1,l2,l3)
                P: set of N points in R^3
                Q: set of M points in R^3
        Check if points in P lie on the boundary of Cl
        Check if points in Q lie on the boundary of Sl
        for (q in Q)
                for(p in P)
                        Aij = sin(|q – p|)/(|q – p|)
                end for
        end for
        for (i = 1 to N)
                for (j = 1 to N)
                        A^TA[i][j] <- 0
                        for (k in M)
                                A^TA[i][j] <- A^TA[i][j] + A^T[i][k] * A[k][j]
                        end for
                end for
        end for
        return A^TA
end BUILD-MATRIX
```

Algorithm Analysis :-

A which is an M X N matrix and A^T will be an N X M matrix. So, A^TA is an N X N matrix and to calculate each element of A^TA, it requires a total of M multiplications and M-1 additions. So, the total no. of operations required for the calculation of entire matrix A^TA are (2M-1) X N^2 . So, the complexity of the algorithm is **$O(N^2M)$.**