G. Nithish
19BCS0012
31.3.2021

Problem Solving:-

1) Quick sort: Program:-

```
#include<bits/stdc++.h>
using namespace std;
Void swap(int *a, int *b)
{int t = *a;
   *a = *b;
   *b = t;
}
int Partition(int arr[], int low, int high)
{
   int pivot= arr[high]; // Pivot
   int i= ((low-1);// Index of smaller element
                    and indicates the rigth
                    Position or pivot found so far
 for (int j=low; j<=high-1; j++)
 {
   if (arr[j]<Pivot)
   { i++;
    Swap (&arr[i], &arr[j]);
   }
 }
```

```
  Swap(&arr[i+1], &arr[high]);
  return(i+1);
}
void quickSort(int arr[], int low, int high)
{
  if(low<high)
  {
    int pi = Partition(arr, low, high);
     quickSort (arr, low, pi-1);
     quickSort (arr, pi+1, high);
    }
  }
void PrintArray(int arr[], int Size)
 {
    int i;
   for (i=0; i<size; i++)
       Cout<< arr[i]<<" ";
      Cout <<endl;
 }
 int main()
{
  int arr[] = {10,7,8,9,1,5};
  int n= Sizeof(arr)/Sizeof(arr[0]);
   Quick Sort (arr, 0, n-1);
```

```
Cout << "Sorted array :\n";
 PrintArray(arr, n);
 return 0;
}
```

Illustration of partition();

arr[] = {10, 80, 30, 90, 40, 50, 70}

Indexes: 0, 1, 2, 3, 4, 5, 6

low=0, high=6, Pivot = arr[h] = 70

Initialize index of Smaller element, i = -1

Traverse elements from j=low to high-1

j=0: Since arr[j] <= Pivot, do i++ and
    Swap(arr[i], arr[j])

i=0

arr[] = {10, 80, 30, 90, 40, 50, 70} //No change as i and j

j=1 : Since arr[j] > pivot, do nothing
 // No change in i and arr[]

j=2 : Since arr[j] <= pivot, do i++ and
    Swap(arr[i], arr[j]

i=1

arr[] = {10, 30, 80, 90, 40, 50, 70}

j=3 : Since arr[j] > pivot do nothing

"No change in i and arr[]

j=4: Since arr[j] <= pivot, do ~~nothing~~ i++ and
swap(arr[i], arr[j])

i=2

arr[]={10, 30, 40, 90, 80, 50, 70}

j=5 : Since arr[j] <= pivot, do i++ and swap
arr[i] with arr[j]

i=3

arr[] = {10, 30, 40, 50, 80, 90, 70}

Finally we place pivot at correct Position by swapping.

arr[i+1] and arr[high] (or Pivot)

arr[] = {10, 30, 40, 50, 70, 90, 80}

Now 70 is at its correct place. All elements smaller than 70 are before it and all element greater than 70 are after it.

2) Explain Matrix multiplication problem with code segment (c/c++/java)

Program

```
#include<stdio.h>

int main(void)

{ int c, d, p, q, m, n, k, tot=0;

  int fst[10][10], sec[10][10], mul[10][10];

 Printf("Please insert the number of row and
          columns for first matrix\n");
 Scanf("%d%d", &m, &n);

Printf("Insert your matrix elements: \n");

  for(c=0; c<m; c++)

    for (d=0; d<n; d++)

      Scanf("%d", &fst[c][d]);
```

```
Printf("Please insert the number of raw
        and columns for second matrix\n");
Scanf("%d %d", &p, &q);
if (n != p)
Printf("Your given matrices can't be
        multiplied with each other.\n");
Else
 { Printf("Insert your element for second
          matrix \n");
   for (c=0; c<p; c++)
     for (d=0; d<q; d++)
       Scanf("%d", &sec[c][d]);
for (c=0; c<m; c++) {
  for (d=0; d<q; d++) {
    for (k=0, k<p; k++) {
      tot = tot + fst[c][k]*sec[k][d];
    }
   mul[c][d] = tot;
    tot =0;
 } }
Printf("The result of matrix multiplication
       or product of the matrices is:\n");

 for (c=0;c<m;c++){

 for (d=0;d<q;d++)

   Printf("%d\t",mul[c][d]);

   Printf("\n");
  }
 }
  return 0;
 }
```

Explanation

↓

Explanation:

Inserting all the elements one by One in Your array needs for loop, followed by a Scanf()

first matrix

```
for (c=0; c<m; c++)
 for (d=0; d<n; d++)
     Scanf("%d", &fst[c][d]);
```

m -> no. of row's n -> no. of columns.

Same as:

second matrix

```
for (c=0; c<p; c++)
for (d=0; d<q; d++)
    Scanf("%d", &sec[c][d]).
```

```
for (c=0; c<m; c++) {
 for (d=0; d<q; d++) {
     for (k=0; k<p; k++) {
  tot = tot + fst[c][k] * sec[k][d]; }
   mul[c][d] = tot;
      tot = 0;
  }  }
```

Here the three loops have been used which Stores the multiplicative value of fst[][] and sec[][] in the variable "tot" and this adding of multiplicative values will continue till it traverses all the values of the array..

At the Same time store every calculated Value of tot in the array mul[][] which will store the resultant multiplication. Now you have to print the resultant 2D array using nested for loop.

```
Printf("The result of matrix multiplication
        or product of the matrices is:\n");
for (c=0; c<g; c++)
   for (d=0; d<bb; d++)
      Printf("%d\t", mul[c][d]);
```

Time complexity

upper bound - $O(n^3)$