Objective:

- Learn to calculate the Time/Space equation.
- Code Smart! [It comes with practice, patience and exposure.]

Task-1: Write time equation of the following code segment.

```
int sum = 0;
for (int k = N; k > 0; k /= 2)
  for (int i = 0; i < k; i++)
    sum++;
```

O(N)

Task-2: You are to develop a function, which finds the sorted position/index (also called as rank) of an element in a given array. The time bound of your function must not be greater than O(N). where 'N' is the size of array.
For example:

Array A = {41, 29, 55, 1, 8, 2, 17}, element = 2
It returns 1

```
int findSortedPosition( int * arr, int N, int key)
{
    int smallerElementsCount=0;

    for (int i=0; i<N; i++)
    {
        if (arr[i]<key)
            smallerElementsCount++;
    }
    return smallerElementsCount;
}
int main()
{
    int a[] = {41, 29, 55, 1, 8, 2, 17};
    cout<<findSortedPosition(a,7,2);
    return 1;
}
```

Task-3: Write a function, which sorts a given array of size 'N' in time bound O(N).
The element of received array always belongs to: {-1, 0, 1}.

```
int findSortedPosition( int * arr, int N, int key)
{
    int smallerElementsCount=0;

    for (int i=0; i<N; i++)
    {
        if (arr[i]<key)
            smallerElementsCount++;
    }
    return smallerElementsCount;
}
```

```
void sortTriArray(int * arr, int N)
{
    int freq[3] = {0,0,0};
    for(int i=0; i<N; i++)
    {
        freq[arr[i]+1]++;
    }
    for(int i=0, k=0; i<3; i++)
    {
        for ( int j=1; j <= freq[i]; j++, k++)
            arr[k] = i-1;
    }
}
int main()
{
    int a[] = {0,0,0,1,-1,1,-1,-1,0,-1,-1};
    sortTriArray(a,11);
    return 1;
}
```

Task-4: Let's define an array as 'Perfect Array'; it is an array in which numbers/data/elements present are in the range of 0 to N-1, where N is the size of array. In this array no number is missing; means all the values in the range 0 to N-1 will be present in the array. But we have a problem, every time we receive an array there is one element missing in it or in other words one element is repeated once. Your task is to find the missing element. Time bound limit is O(N).

```
int findMissingElement(int * arr, int N)
{
    for (int i = 0; i < N; i++)
    {
        if (arr[i] != i)
            swap(arr[i], arr[arr[i]]);
    }
    int j=0;
    while(j<N && arr[j]==j)
    {
        j++;
    }

    return j!=N? j: -1;
}
int main()
{
    int a[] = {3,0,2,1,0};
    cout<<findMissingElement(a,5);
    return 1;
}
```

Task-5: Given an array of size 'N' consisting of unique integers. Your task is to write code in O(logN) time bound to find such an index 'i' on which the following property holds:

arr[i-1] > arr[i] < arr[i+1].

```
int findMidMinimum( int * arr, int N )
```

```
{
    if (N==1)
        return arr[0];
    int lb=1, ub=N-2;
    int mid;
    while(lb<=ub)
    {
        mid = lb + (ub-lb)/2;

        if (arr[mid-1] < arr[mid] && arr[mid] < arr[mid+1])
            ub = mid-1;
        else if (arr[mid-1] > arr[mid] && arr[mid] > arr[mid+1])
            lb = mid+1;
        else if (arr[mid-1] > arr[mid] && arr[mid] < arr[mid+1])
            return arr[mid];
        else
            ub = mid - 1;//lb = mid+1;//5, 6, 9, 8, 10, 12
    }
    if(mid==1)
        return arr[mid-1];
    else if (mid==N-2)
        return arr[mid+1];
    return -1;
}
int main()
{
    int a[] = {6, 5, 4, 13, 8, 10, 12};//{4, 5, 6, 9, 8, 10,
12};//{15,16,4,3,5};//{5,2,3,4,5};//{5,4,3,2,1};//{ 5, 6, 9, 8, 10,
12};
    cout<<findMidMinimum(a, 5);
    return 1;
}
```

Task-6: Write a O(logN) time bound function, which receives a sorted (ascending) array of size 'N' consisting of distinct integers. Your function finds such an index 'i' such that A[i] = i else return -1.

```
int findIndexEqualValue( int * arr, int N)
{
    int lb = 0, ub = N-1, mid;
    while(lb<=ub)
    {
        mid = lb + (ub-lb)/2;
        if (arr[mid]==mid)
            return mid;
        else if (arr[mid]>mid)
            ub = mid - 1;
        else if (arr[mid]<mid)
            lb = mid + 1;
    }
    return -1;
}
int main()
{
```

```
    int a[] = {-10,-9,-8,-7,4,56};
    cout<<findIndexEqualValue(a, 6);
    return 0;
}
```

Task-7: Write the function, which receives an N order matrix and calculate the sub matrix with largest possible sum. Obviously the matrix elements can be negative as well.
Note: You may see different versions of this problem in different programming competition.

```
int largestSubMatrixSum(int (*m)[3], int N)
{
    int largestSum = m[0][0];
    int sum;
    for (int i1 = 0; i1 < N; i1++)
    {
        for (int j1=0; j1 < N; j1++)
        {
            for (int i2 = 0; i2 < N; i2++)
            {
                for (int j2=0; j2 < N; j2++)
                {
                    sum=0;
                    for (int r=i1; r<=i2; r++)
                    {
                        for (int c=j1; c<=j2; c++)
                        {
                            sum = sum + m[r][c];
                        }
                    }
                    if (sum > largestSum)
                        largestSum = sum;
                }
            }

        }
    }
    return largestSum;
}
```

How to submit?

1. For each task create a separate .cpp file and write your code/solution in it.
   a. E.g. task1.cpp will contain time equation
   b. task2.cpp will contain solution of Task-2
2. Put all your code file(s) in a folder named as rollno-Assignment-1. Example folder name bsef12m001-Assignment-1.
3. Compress the folder.
4. Send the compressed folder to course email with email subject as rollno-Assignment-1.
5. If you have any queries/confusions in submitting the assignment then feel free to discuss but discuss in time.

Continuous effort - not strength or intelligence - is the key to unlocking our potential.
… Liane Cordes …