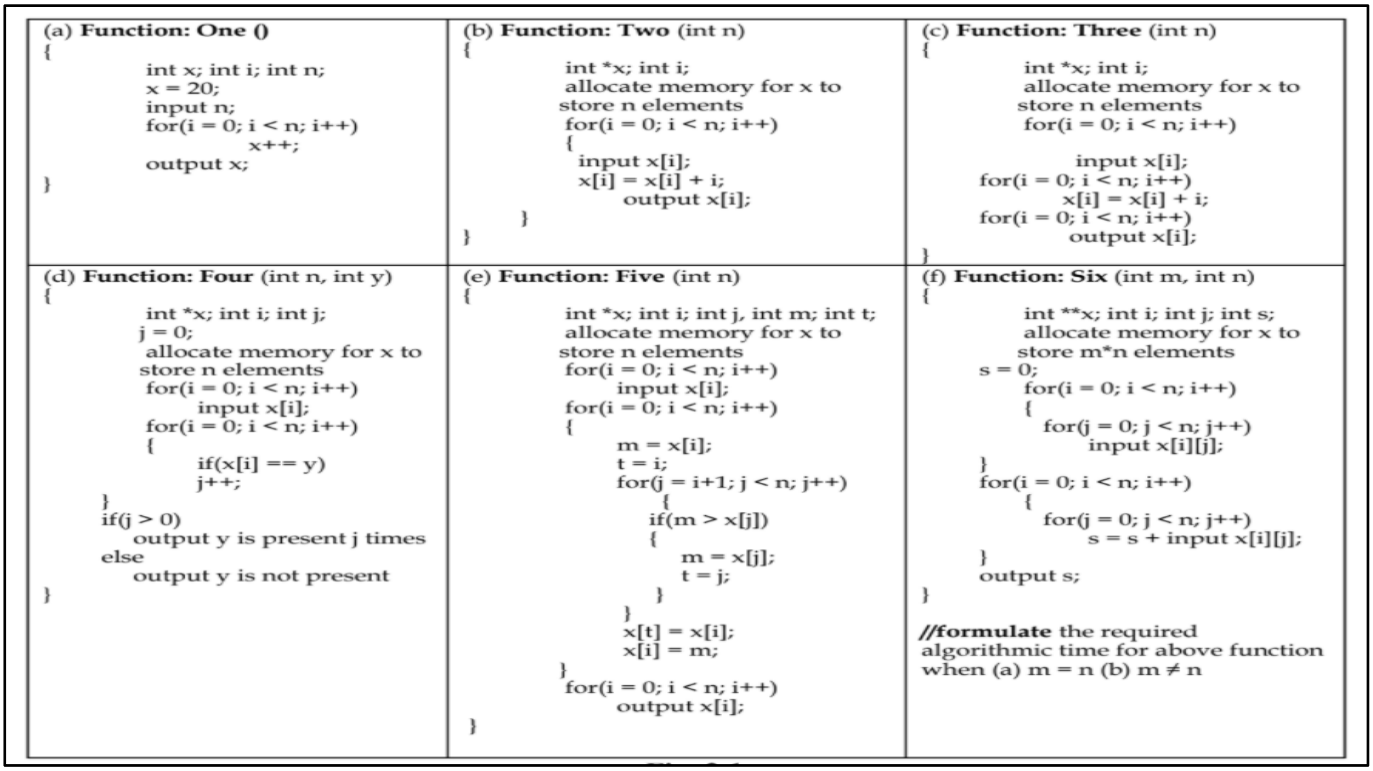
Algorithms and Problem-Solving Lab (15B17CI471)
EVEN 2022
Week -2 (14 Feb - 19 Feb 2022)

Q1. You are given an array A[m] where first n cells contain integers in sorted order and the rest of the cells are filled with 0. Here assumes m>>n and value of n is unknown. Implement an algorithm that takes an integer x as input and finds a position in the array containing x, if such a position exists, in O (log n) time.

Q2. Assume that we are given *n* pairs of items as input, where the first item is a number and the second item is one of three colours (red, blue, or yellow). Further assume that the items are sorted by number. Give an *O*(*n*) algorithm to sort the items by colour (all reds before all blues before all yellows) such that the numbers for identical colours stay sorted. For example: (1, blue), (3, red), (4, blue), (6, yellow), (9, red) should become (3, red), (9, red), (1, blue), (4, blue), (6, yellow).

Q3. Find the complexity of the following code snippets:



(a) **Function: One ()**

```
{
        int x; int i; int n;
        x = 20;
        input n;
        for(i = 0; i < n; i++)
                x++;
        output x;
}
```

(b) **Function: Two** (int n)

```
{
        int *x; int i;
        allocate memory for x to
       store n elements
        for(i = 0; i < n; i++)
        {
         input x[i];
         x[i] = x[i] + i;
              output x[i];
     }
}
```

(c) **Function: Three** (int n)

```
{
        int *x; int i;
        allocate memory for x to
       store n elements
        for(i = 0; i < n; i++)

              input x[i];
    for(i = 0; i < n; i++)
             x[i] = x[i] + i;
    for(i = 0; i < n; i++)
             output x[i];
}
```

(d) **Function: Four** (int n, int y)

```
{
        int *x; int i; int j;
       j = 0;
        allocate memory for x to
       store n elements
        for(i = 0; i < n; i++)
              input x[i];
        for(i = 0; i < n; i++)
        {
              if(x[i] == y)
              j++;
    }
    if(j > 0)
        output y is present j times
    else
        output y is not present
}
```

(e) **Function: Five** (int n)

```
{
        int *x; int i; int j, int m; int t;
        allocate memory for x to
       store n elements
        for(i = 0; i < n; i++)
              input x[i];
        for(i = 0; i < n; i++)
        {
              m = x[i];
              t = i;
              for(j = i+1; j < n; j++)
                {
                 if(m > x[j])
                 {
                     m = x[j];
                     t = j;
                 }
               }
               x[t] = x[i];
               x[i] = m;
       }
        for(i = 0; i < n; i++)
              output x[i];
 }
```

(f) **Function: Six** (int m, int n)

```
{
        int **x; int i; int j; int s;
        allocate memory for x to
       store m*n elements
    s = 0;
        for(i = 0; i < n; i++)
        {
          for(j = 0; j < n; j++)
              input x[i][j];
    }
    for(i = 0; i < n; i++)
        {
          for(j = 0; j < n; j++)
              s = s + input x[i][j];
    }
    output s;
}

//formulate the required
algorithmic time for above function
when (a) m = n (b) m ≠ n
```

Q4. Implement the recursive algorithms for (a) Tower of Hanoi and (b) Fibonacci Number computation and analyse the space and time requirements of both the algorithms.

Q5. Implement the algorithm (Algo_1) presented below and discuss which task this algorithm performs. Also, analyse the time complexity and space complexity of the given algorithm. Further, implement the algorithm with following modification: replace $m = \lceil 2n/3 \rceil$ with $m = \lfloor 2n/3 \rfloor$, and compare the tasks performed by the given algorithm and modified algorithm.

```
Algo_1(A [0 ... n-1])
{
    if n = 2 and A[0] > A[1]
        swap A[0] ↔ A[1]
    else if n > 2
        $m = \lceil 2n/3 \rceil$
        Algo_1 (A [0 .. m − 1])
        Algo_1 (A [n − m .. n − 1])
        Algo_1 (A [0 .. m − 1])
}
```