Practice Questions

Solution 1:-

```
class Solution {
    public int editDistance(String s1, String s2) {
        // Code here
       int m=s1.length();
       int n=s2.length();
           int dp[][]=new int[m+1][n+1];
        for (int[] row : dp) {
       Arrays.fill(row, -1);
    }

       return ed(s1,s2,m,n,dp);



    }
    static int ed(String s1,String s2,int m,int n,int dp[][]){

        if(dp[m][n]!=-1){
            return dp[m][n];
        }
        if(m==0)
        return n;
        if(n==0)
        return m;
        if(s1.charAt(m-1)==s2.charAt(n-1)){
          dp[m][n]= ed(s1,s2,m-1,n-1,dp);
        }
        else{
            //if the character are not matchinge then we can perform the
three operation
            //one insertion and other deletion and other replacmenet
```

```
        dp[m][n]=1+Math.min(Math.min(ed(s1,s2,m,n-
1,dp),ed(s1,s2,m-1,n,dp)),ed(s1,s2,m-1,n-1,dp));
      }
      return dp[m][n];
   }
}
```

Solution 2:-

```
class Solution
{
   int atoi(String s) {
      int ans=0;
      boolean flag=false;

      for(int i=0;i<s.length();i++){
         char c=s.charAt(i);
         if(i==0 && c=='-'){
            continue;
         }else if(Character.isDigit(c)==false){
            flag=true;
            break;
         }else{
            int temp=c-'0';
            ans*=10;
            ans+=temp;
         }
      }
```

```
        if(flag) return -1;
        if(s.charAt(0)=='-') ans*=-1;
        return ans;
    }
}
```

Solution 3:-

```
class Solution
{
  static int isCircle(int n, String a[])
  {
  ArrayList<ArrayList<Integer>> adj=new
ArrayList<ArrayList<Integer>>();
  for(int i=0;i<26;i++)
  {
     adj.add(new ArrayList<Integer>());
  }
   int in[]=new int[26];
  int out[]=new int[26];
  for(int i=0;i<n;i++)
  {
     String temp=a[i];
     adj.get(temp.charAt(0)-'a').add(temp.charAt(temp.length()-1)-
'a');
     out[temp.charAt(0)-'a']++;
     in[temp.charAt(temp.length()-1)-'a']++;
  }
```

```
  for(int i=0;i<26;i++)
  {
     if(in[i]!=out[i])
     return 0;
  }
  boolean visited[]=new boolean[26];
  dfs(adj,a[0].charAt(0)-'a',visited);
  for(int i=0;i<26;i++)
  {
     if(visited[i]==false && (in[i]!=0 || out[i]!=0))
     return 0;
  }
  return 1;
  }
  static void dfs(ArrayList<ArrayList<Integer>>adj,int ind,boolean
visited[])
  {
     visited[ind]=true;
     for(int i:adj.get(ind))
     {
        if(visited[i]==false)
        {
           dfs(adj,i,visited);
        }
     }
  }
}
```

Solution 4:-

```
class Solution{
  //Function to return a tree created from postorder and inoreder traversals.
  Node buildTree(int in[], int post[], int n) {
    HashMap<Integer,Integer> map=new HashMap<>();

    for(int i=0;i<in.length;i++)
    {
       map.put(in[i],i);
    }

    Node ans=construct(post,0,n-1,in,0,n-1,map);
    return ans;
 }

 Node construct(int [] post,int poststart,int postend,int [] in,int instart,int inend, HashMap<Integer,Integer> map)
 {
    if(poststart>postend || instart>inend)
    {
       return null;
    }

    Node newNode=new Node(post[postend]);
    int pos=map.get(post[postend]);
    int noc=pos-instart;
    newNode.left=construct(post,poststart,poststart+noc-1,in,instart,pos-1,map);
    newNode.right=construct(post,poststart+noc,postend-1,in,pos+1,inend,map);
```

```
        return newNode;
    }
}
```