数据结构中，沿着某条搜索路线，依次对树中每个结点均做一次且仅做一次访问。对二叉树的结点从 1 开始进行连续编号，要求每个结点的编号大于其左、右孩子的编号，同一结点的左右孩子中，其左孩子的编号小于其右孩子的编号，可采用（ ）次序的遍历实现编号。
正确答案: C 你的答案: 空 (错误)
先序
中序
后序
从根开始按层次遍历

完全二叉树是指深度为 K 的，有 n 个结点的二叉树，当且仅当其每一个结点都与深度为 K 的满二叉树中编号从 1 至 n 的结点一 一对应将一棵有 50 个结点的完全二叉树按节点编号，如根节点的编号为 1，那么编号为 25 的结点是（ ）?
正确答案: B 你的答案: 空 (错误)
无左、右孩子
有左孩子，无右孩子
有右孩子，无左孩子
有左、右孩子

小易给定你数字[illegible]和系数[illegible]。每次操作你可以将[illegible]变成[illegible]或者将[illegible]变成[illegible]。问至少几次操作使得[illegible]。

```
import java.util.*;

public class Main {

  public static void main(String[] args){
    Scanner input=new Scanner(System.in);
    int n=input.nextInt();
    input.nextLine();
    for(int i=0;i<n;i++){
      String[] s = input.nextLine().split(" ");
      int A=Integer.parseInt(s[0]);
      int B=Integer.parseInt(s[1]);
      int p=Integer.parseInt(s[2]);
      int q=Integer.parseInt(s[3]);
      System.out.println(count(A,B,p,q,0));
    }
  }

  private static int count(double A,double B,double p,double q,int num){
    if(B-A<=p)
      return num+1;
    else if((B-A)/p<=q)
      return num+2;
```

```
        else {
            return count(A,B,p*q*q,q,num+2);
        }
    }

}
```

小易给定你一个长度为n的正整数序列a，你每次可以使用k的代价将某个数加一或者减一，你希望用最少的代价使得所有数的乘积等于x，求最小代价（操作结束后每个数也必须是正整数）。

```
import java.util.Scanner;
public class Main{

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int target = scanner.nextInt();

        int[] h = new int[n];
        for(int i = 0; i < n; i++) {
            h[i] = scanner.nextInt();
        }

        //dp[i][j] 表示前 i 个数字 使得为 j 所需要的代价
        Integer[][] dp = new Integer[n+1][target + 1];

        int sum = 0;
        for(int i = 1; i <= n; i++) {
            sum += h[i-1] - 1;
            dp[i][1] = sum;
        }

        for (int j = 1; j <= target; j++) {
            dp[1][j] = Math.abs(j - h[1-1]);
        }


        int result = doCheck(dp, n, target, h);

        System.out.println(result);

    }

    private static int doCheck(Integer[][] dp, int n, int target, int[] h) {
```

```
        if (dp[n][target] != null) {
            return  dp[n][target];
        }

        //dp[n][target] == null

        int min = Integer.MAX_VALUE;
        for (int i = 1; i <= target; i++) {
            if (target % i != 0) {
                continue;
            }

            int j = target / i;
            min = Math.min(min, doCheck(dp, n-1, j, h) + Math.abs(h[n-1] - i));
        }



        return dp[n][target] = min;
    }
}
```

小易有n根柱子，第i根柱子的高度为h[i]。一开始小易站在第一根柱子上。小易能从第i根柱子跳到第j根柱子，当且仅当j<i且h[j]<=h[i]。其中k为指定的一个数字。
另外小易拥有一次释放超能力的机会。这个超能力能让小易从柱子i跳到任意满足j>i的柱子j而无视柱子高度的限制。
现在小易想道，小易是否能到达第n根柱子。

```
import java.util.Scanner;

public class Main{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int T = sc.nextInt();
        int n = 0, k = 0;
        for (int i = 0; i < T; i++) {
            n = sc.nextInt();
            k = sc.nextInt();
            int[] nums = new int[n];
            for (int j = 0; j < n; j++)
                nums[j] = sc.nextInt();
            System.out.println(solution(n, k, nums));
        }
    }
```

```
    public static String solution(int n, int k, int[] nums) {
        int big = 1;
        int index = 0;
        while (index < nums.length - 1) {
            int tmp = index;
            int max = 0, max_index = index;
            for (int j = index + 1; j < index + 1 + k && j < nums.length; j++) {
                if (nums[j] < nums[index]) {
                    max_index = (max > nums[j]) ? max_index : j;
                    max = Math.max(nums[j], max);
                }
            }
            index = max_index;
            if (tmp == index && big > 0) {
                big--;
                max = 0;
                max_index = index;
                for (int j = index + 1; j < index + 1 + k && j < nums.length; j++) {
                    max_index = (max > nums[j])? max_index : j;
                    max = Math.max(nums[j], max);
                }
                index = max_index;
            }
            else if (tmp == index && big <= 0)
                return "NO";
        }
        return "YES";
    }
}
```

小易的公司一共有 n 名员工, 第 i 个人每个月的薪酬是 i 万元。
现在小易的老板向小易提了 m 次询问, 每次询问老板都会给出一个整数 x, 小易要快速回答老板工资等于 x 的员工的数量。

```
import java.util.HashMap;
import java.util.Scanner;
public class Main{
    public static void main(String[] args){
        int n,m;
        int input;
        Scanner sc = new Scanner(System.in);
        n=sc.nextInt();
        m=sc.nextInt();
        int a;
        HashMap<Integer,Integer> hm = new HashMap();
        for(int i=0;i<n;i++){
```

```
            a=sc.nextInt();
            if(hm.containsKey(a)){
                hm.put(a,hm.get(a)+1);
            }else{
                hm.put(a,1);
            }
        }
        while(m--!=0){
            input=sc.nextInt();
            if(hm.get(input)!=null){
                System.out.println(hm.get(input));
            }else{
                System.out.println(0);
            }
        }

    }
}
```