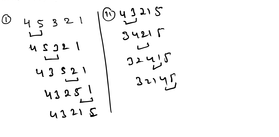
```
Cars[] arr = new Cars[5];
```

```
public static void Sort(int[] arr) {
    for (int turn = 1; turn < arr.length; turn++) {
        for (int i = 0; i < arr.length - turn; i++) {
            if (arr[i] > arr[i + 1]) {
                int temp = arr[i];
                arr[i] = arr[i + 1];
                arr[i + 1] = temp;
            }
        }
    }
}
```

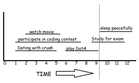


```
@Override
public int compareTo(Cars o) {
    // TODO Auto-generated method stub
    return 0;
}
```

```
public static <T extends Comparable<T>> void Sort(T[] arr) {
    for (int turn = 1; turn < arr.length; turn++) {
        for (int i = 0; i < arr.length - turn; i++) {
            if (arr[i].compareTo(arr[i + 1]) > 0) {
                T temp = arr[i];
                arr[i] = arr[i + 1];
                arr[i + 1] = temp;
            }
        }
    }
}
```

```
        arr[0] = new Cars(200, 10, "White");// P S C
        arr[1] = new Cars(1000, 20, "Black");
        arr[2] = new Cars(345, 3, "Yellow");
        arr[3] = new Cars(34, 89, "Grey");
        arr[4] = new Cars(8907, 6, "Red");
```

You are actually very busy man. You have a big schedule of activities. Your aim is to do as much as activities as possible.



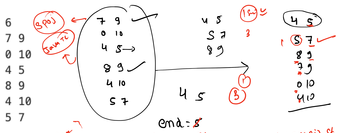
In the given figure, if you go to date with crush, you cannot participate in the coding contest and you can't watch the movie. Also if you play DotA, you can't study for the exam. If you study for the exam you can't sleep peacefully. The maximum number of activities that you can do for this schedule is 3.

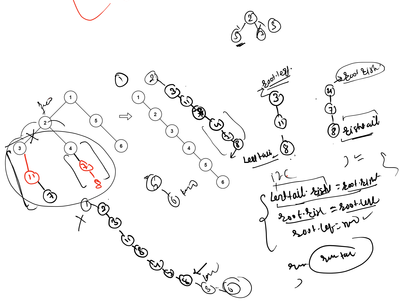
Either you can

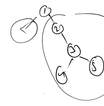
- watch movie, play DotA and sleep peacefully (or)
- date with crush, play DotA and sleep peacefully

Input

The first line consists of an integer **T**, the number of test cases. For each test case the first line consists of an integer **N**, the number of activities. Then the next **N** lines contains two integers **m** and **n**, the start and end time of each activity.



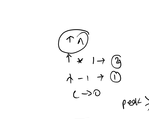




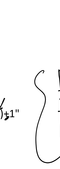
```
public void flatten(TreeNode root) {
        MakeLL(root);
    }
    public TreeNode MakeLL(TreeNode root){

      if (root == null) {
          return null;
      }
      if (root.left == null && root.right == null) {
          return root;
      }

        TreeNode left_tail=MakeLL(root.left);
        TreeNode right_tail=MakeLL(root.right);
      if(left_tail!=null){
          left_tail.right=root.right;
          root.right=root.left;
          root.left=null;
      }
      return right_tail!=null ? right_tail : left_tail;
    }
```



```
public static void main(String[] args) {
        // TODO Auto-generated method stub
        String str="2-3/(5*2)+1";
        String s="";
        Stack<Character> st = new Stack<>();
        for (int i = 0; i < str.length(); i++) {
            char ch =str.charAt(i);
            if(ch>='0'&& ch<='9') {
                s+=ch;
            }
            else if(ch=='(') {
                st.push(ch);
            }
            else if(ch==')') {
                while(st.peek()!='(') {
                    s=s+st.pop();
                }
                st.pop();//'('
            }
            else {
                while(!st.isEmpty() && getvale(st.peek())>=getvale(ch)) {
                    s=s+st.pop();
                }
                st.push(ch);
            }
        }
    }
```



String [ ] arr= ["100", "200", "+", "2", "/", "5", "*", "2", "+"]

String [ ]arr = ["2", "3", "1", "*", "+", "9", "-"]