代码随想录—ACM输入输出模板

作者：程序员Carl

前言

很多录友苦于不熟悉 ACM 输入输入结构，在笔试和面试的时候，处理数据输入输出就花费了大量的时间，以至于算法题没写完，甚至是 根本就写不对输入输出的方式。

下面，我针对常见的25种 ACM输入与输出方式，给大家总结了模板写法，包括了C++、Java、Python、Go、JS等主流编程语言。

大家可以拿去直接“背诵”。

每种输入输出，都配套的对应的卡码网练习题，注意本PDF只给出每种情况的ACM输入输出的模板写法，没有题目的完整代码，想看题目完整代码可以看卡码网题解区

1. 多行输入，每行两个整数

练习题：A+B问题I

C++

```
#include<iostream>
using namespace std;
int main() {
    int a, b;
    while (cin >> a >> b) cout << a + b << endl;
}
```

Java

```
import java.lang.*;
import java.util.*;

public class Main{
    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        while(in.hasNextInt()){
            int a = in.nextInt();
            int b = in.nextInt();
            System.out.println(a+b);
        }
    }
}
```

Python

```python
import sys
# 接收输入
for line in sys.stdin:
    a, b = line.split(' ')
    # 输出
    print(int(a) + int(b))
    # 输出换行
    print()
```

Go

```go
package main

import "fmt"

func main(){
    var a, b int
    for {
        _, err := fmt.Scanf("%d %d",&a, &b)
        if err != nil {
            break
        }
        fmt.Println(a + b)
    }
}
```

JavaScript（Node）

```javascript
// 引入readline模块来读取标准输入
const readline = require('readline');

// 创建readline接口
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

// 处理输入和输出
function processInput() {
  rl.on('line', (input) => {
    // 将输入按空格分割成a和b的数组
    const [a, b] = input.split(' ').map(Number);

    // 计算a和b的和并输出
    const sum = a + b;
    console.log(sum);
  });
}

// 开始处理输入
processInput();
```

2. 多组数据，每组第一行为n, 之后输入n行两个整数

练习题：A+B问题II

C++

```
#include<iostream>
using namespace std;
int main() {
    int n, a, b;
    while (cin >> n) {
        while (n--) {
            cin >> a >> b;
            cout << a + b << endl;
        }
    }
}
```

Java

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        while (scanner.hasNext()) {
            int n = scanner.nextInt();
            while (n-- > 0) {
                int a = scanner.nextInt();
                int b = scanner.nextInt();
                System.out.println(a + b);
            }
        }
    }
}
```

Python

```python
while 1:
    try:
        N = int(input())
        for i in range(N):
            l = list(map(int,input().split()))
            print(sum(l))
    except:
        break
```

Go

```go
package main

import "fmt"

func main() {
  var n, a, b int
  for {
    _, err := fmt.Scan(&n)
    if err != nil {
      break
    }
    for n > 0 {
      _, err := fmt.Scan(&a, &b)
      if err != nil {
        break
      }
      fmt.Println(a + b)
      n--
    }
  }
}
```

JS

```javascript
// 引入readline模块来读取标准输入
const readline = require("readline");

// 创建readline接口
const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout,
});

function processInput() {
    let pathArr = [];
    rl.on("line", (input) => {
        let path = input.split(" ").map(Number);
        // 将输入转为数组，根据length属性判断输入数字的个数
        if (path.length == 1) {
            pathArr.push(path);
        } else {
            const [a, b] = path
            const sum = a + b;
            console.log(sum);
        }
    });

}
processInput();
```

3. 若干行输入，每行输入两个整数，遇到特定条件终止

练习题：A+B问题III

C++

```
#include<iostream>
using namespace std;
int main() {
    int a, b;
    while (cin >> a >> b) {
        if (a == 0 && b == 0) break;
        cout << a + b << endl;
    }
}
```

Java

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        while (scanner.hasNext()) {
            int a = scanner.nextInt();
            int b = scanner.nextInt();
            if (a == 0 && b == 0) {
                break;
            }
            System.out.println(a + b);
        }
    }
}
```

Python

```python
import sys

while True:
    s = input().split() # 一行一行读取
    a, b = int(s[0]), int(s[1])
    if not a or not b: # 遇到 0, 0 则中断
        break
    print(a + b)
```

Go

```go
package main

import "fmt"

func main() {
  var a, b int
  for {
    _, err := fmt.Scan(&a, &b)
    if err != nil {
      break
    }
    if a == 0 && b == 0 {
      break
    }
    fmt.Println(a + b)
  }
}
```

JavaScript

```javascript
// 引入readline模块来读取标准输入
const readline = require('readline');

// 创建readline接口
const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

function preoceeInput() {
    rl.on('line', (input) => {
        const [a, b] = input.split(' ').map(Number);
        // # 遇到 0, 0 则中断
        if (a === 0 && b === 0) {
            return;
        } else {
            console.log(a + b);
        }
    });
}

preoceeInput()
```

```
// 使用 Node.js 的 readline 模块来模拟 C++ 中的 cin 和 cout 操作
function main() {
    // 导入readline模块
    const readline = require('readline');

    // 创建readline接口
    const rl = readline.createInterface({
        input: process.stdin,  // 从标准输入读取数据
        output: process.stdout  // 将输出写入标准输出
    });

    // 监听用户的输入事件
    rl.on('line', (input) => {
        // 将输入拆分为两个数，并将其转换为数字
        const [a, b] = input.split(' ').map(Number);

        // 判断输入的两个数是否都为0
        if (a === 0 && b === 0) {
            rl.close();  // 如果是，则关闭输入流，结束程序
        } else {
            console.log(a + b);  // 否则，计算并输出两数之和
        }
    });
}

main();  // 调用主函数开始程序的执行
```

4. 若干行输入，遇到0终止，每行第一个数为N，表示本行后面有N个数

练习题：4. A + B问题IV

C++

```cpp
#include<iostream>
using namespace std;
int main(){
    int n, a;
    while (cin >> n) {
        if (n == 0) break;
      // 计算累加值
        int sum = 0;
        while (n--) {
            cin >> a;
            sum += a;
        }
        cout << sum << endl;
    }
}
```

Java

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        while (scanner.hasNext()) {
            int n = scanner.nextInt();
            if (n == 0) {
                break;
            }
            int sum = 0;
            for (int i = 0; i < n; i++) {
                sum += scanner.nextInt();
            }
            System.out.println(sum);
        }

    }
}
```

Python

```
import sys

for line in sys.stdin:
    nums = line.split()
    nums = list(map(int, nums))
    n = nums[0]
    if not n:
        break
    print( sum(nums[-n:]) )
```

Go

```
package main

import "fmt"

func main() {
  var n, a int
  for {
    _, err := fmt.Scan(&n)
    if err != nil {
      break
    }
    if n == 0 {
      break
    }
    sum := 0
    for n > 0 {
      _, err := fmt.Scan(&a)
      if err != nil {
        break
      }
      sum += a
      n--
    }
    fmt.Println(sum)
  }
}
```

JavaScript(Node)

```
// 引入readline模块来读取标准输入
const readline = require('readline');

// 创建readline接口
const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

function preoceeInput() {
    rl.on('line', (input) => {
        // 读入每行数据，将其转换为数组
        const line = input.split(' ').map(Number);
        // 判断读入的第一个数字是否为0
        if (line[0] === 0) {
            return;
        } else {
            let sum = 0;
            for (let i = 1; i < line[0] + 1; i++) {
                sum += line[i];
            }
            console.log(sum);
        }
    });
}

preoceeInput()
```

5. 若干行输入，每行包括两个整数a和b，由空格分隔，每行输出后接一个空行。

练习题：A+B问题 VII

C++

```
#include<iostream>
using namespace std;
int main() {
    int a, b;
    while (cin >> a >> b) cout << a + b << endl << endl;
}
```

Java

```java
import java.util.Scanner;
public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        while(sc.hasNextLine()){
            int a = sc.nextInt();
            int b = sc.nextInt();
            System.out.println(a + b);
            System.out.println();
        }
    }
}
```

Python

```python
while True:
    try:
        x, y = map(int, (input().split()))
        print(x + y)
        print()
    except:
        break
```

Go

```go
package main

import "fmt"

func main() {
  var a, b int
  for {
    _, err := fmt.Scan(&a, &b)
    if err != nil {
      break
    }
    fmt.Printf("%d\n\n", a+b)
  }
}
```

Js

```
const readline=require('readline');
const rl=readline.createInterface({
    input:process.stdin,
    output:process.stdout
});
function Sum(){
    rl.on("line",(input)=>{
        const [a,b]=input.split(' ').map(Number);
        console.log(a+b+"\n");
    });
}
Sum();
```

6. 多组n行数据，每行先输入一个整数N，然后在同一行内输入M个整数,每组输出之间输出一个空行。

练习题：A+B问题 VIII

C++

```
#include<iostream>
using namespace std;
int main() {
    int n, m, a;
  // 输入多组数据
    while (cin >> n) {
      // 每组数据有n行
       while(n--) {
         cin >> m;
         int sum = 0;
         // 每行有m个
         while(m--) {
           cin >> a;
           sum += a;
         }
         cout << sum << endl;
         cout << endl;
       }
    }
}
```

Java

```
import java.util.Scanner;
public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        while(sc.hasNextLine()){
            int N = sc.nextInt();
          // 每组有n行数据
            while(N-- > 0){
                int M = sc.nextInt();
                int sum = 0;
              // 每行有m个数据
                while(M-- > 0){
                    sum += sc.nextInt();
                }
                System.out.println(sum);
                if(N > 0) System.out.println();
            }
        }

    }
}
```

Python

```
while 1:
    try:
        N = int(input())
        for i in range(N):
            n = list(map(int, input().split()))
            if n[0] == 0:
                print()
                continue
            print(sum(n[1:]))
            if i<N-1:
                print()
    except:
        break
```

Go

```
package main

import "fmt"

func main() {
    var N, M, num, sum int

    // 无限循环，直到没有更多的输入数据
    for {
        // 尝试读取一组测试数据的数量
        if _, err := fmt.Scan(&N); err != nil {
            break // 如果读取失败（例如，输入结束），则退出循环
        }

        // 处理每一组测试数据
        for i := 0; i < N; i++ {
            sum = 0 // 初始化当前组的数字和为0
            fmt.Scan(&M) // 读取当前组的数字数量

            // 读取并累加当前组的所有数字
            for j := 0; j < M; j++ {
                fmt.Scan(&num) // 读取一个数字
                sum += num    // 将该数字加到总和中
            }

            // 输出当前组的数字和
            fmt.Println(sum)

            // 如果不是当前组的最后一个数字，输出一个空行
            if i < N-1 {
                fmt.Println()
            }
        }
    }
}
```

JavaScript

```
// 引入readline模块来读取标准输入
const readline = require('readline');

// 创建readline接口
const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

function preoceeInput() {
    let n;
    rl.on('line', (input) => {
        // 读入每行数据，将其转换为数组
        const readline = input.split(' ').map(Number);
        if (readline.length === 1) {
                n = readline[0];
            } else {
                let sum = 0;
                for (let i = 1; i < readline[0] + 1; i++) {
                    sum += readline[i];
                }
                if (n > 1) {
                    console.log(sum + "\n");
                    n--;
                } else { // 如果是第n行 只输出结果不换行
                    console.log(sum);
                }
            }
        });
}

preoceeInput()
```

7. 多组测试样例，每组输入数据为字符串，字符用空格分隔,输出为小数点后两位

练习题：7. 平均绩点

C++

```
#include <iostream>
using namespace std;
int main() {
    string s;
    while (getline(cin, s)) { // 接受一整行字符串
      for(int i = 0; i < s.size();i++) { // 遍历字符串

      }
    }
}
```

```
#include <iostream>
#include <stdio.h>
int main() {
  float sum = 10.0;
  int count = 4;
  printf("%.2f\n", sum / count);
}
```

Java

```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        while (in.hasNextLine()) {
            String line = in.nextLine(); // 接收一整行字符串作为输入
            String[] items = line.split(" "); // 字符串分割成数组
            for (String item : items) { // 遍历数组

            }
        }
    }


}
```

```java
public class Main {
    public static void main(String[] args) {
        double avg = 3.25;
        System.out.printf("%.2f\n", avg);
    }
}
```

Python

```python
while 1:
    try:
        n = input().replace(" ", "").replace("A", "4").replace("B", "3").replace("C",
"2").replace("D", "1").replace(
            "F", "0")
        s = 0
        for i in n:
            if i not in '43210':
                print('Unknown')
                break
            s += int(i)
        else:
            print(f"{s / len(n):.2f}")
    except:
        break
```

Go

```
package main

import(
    "bufio"
  "fmt"
  "os"
  "strings"
)

func main(){
    //创建一个bufio.Reader对象，用于从标准输入（即键盘）读取数据
    reader := bufio.NewReader(os.Stdin)
    for{
        s, _, err := reader.ReadLine()
        s_list := strings.Split(string(s), " ")
        //如果err的值不等于nil，则表示输入结束
        if err != nil{
            break
        }
        for i := 0; i < len(s_list); i++{

        }
    }
}
```

```
fmt.Println(fmt.Sprintf("%.2f", 3.14159))
```

JavaScript(Node)

```
// 引入readline模块读取输入
const readline = require("readline");
// 创建readline接口
const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout,
});

function processInput() {

    rl.on("line", (input) => {
        let arr = input.split(" ");
        // 遍历
        for (let i = 0; i < arr.length; i++) {
        }
    });
}
processInput();
```

```
let svg = 3.14159
console.log(svg.toFixed(2))
```

8. 多组测试用例，第一行为正整数n, 第二行为n个正整数，n=0时，结束输入，每组输出结果的下面都输出一个空行

练习题 8. 摆平积木

C++

```
#include<iostream>
#include<vector>
using namespace std;

int main() {
    int n;
    while (cin >> n) {
        if (n == 0) break;
      // 创建vector
        vector<int> nums = vector<int>(n, 0);
        // 输入一组数据
        for (int i = 0; i < n; i++) {
            cin >> nums[i];
        }
        // 遍历
        for (int i = 0; i < n; i++) {
           cout << nums[i];
        }
        cout << result << endl;
        cout<< endl;
    }
}
```

Java

```
import java.util.ArrayList;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        while (scanner.hasNext()) {
            Integer size = scanner.nextInt();
            if (size == 0) {
                break;
            }
          // 创建list
            ArrayList<Integer> list = new ArrayList<>();
          // 添加一组数据到list中
            for (int i = 0; i < size; i++) {
                int num = scanner.nextInt();
                list.add(num);
            }
            // 遍历
            for (int i = 0; i < list.size(); i++) {
                System.out.println(list.get(i));
            }
            System.out.println(res);
            System.out.println();
        }
    }
}
```

Python

```
while 1:
    try:
        n = int(input())
        if n == 0:
            break
        ls = list(map(int, input().split()))
        # 遍历
        for i in ls:
            #操作
        print(moves//2)
        print()
    except:
        break
```

Go

```go
package main

import (
    "fmt"
)

func main() {
    var n int
    for {
        _, err := fmt.Scanf("%d", &n)
        if err != nil || n == 0 {
            break
        }
        nums := make([]int, n)

        for i := 0; i < n; i++ {
          // 读取一个整数，存放在数组中
            fmt.Scanf("%d", &nums[i])
        }
        for i := 0; i < n; i++ {
            fmt.Println(nums[i])
        }
        fmt.Println(result)
        fmt.Println()
    }
}
```

Js

```javascript
const rl = require("readline").createInterface({ input: process.stdin });
var iter = rl[Symbol.asyncIterator]();
const readline = async () => (await iter.next()).value;

void (async function () {

    while ((line = await readline())) {
        // 读取输入数组长度
        let size = parseInt(line)
        if(size == 0) break
        let input = await readline()
        // 将输入的数字转换成整数数组
        let arr = input.split(" ").map(Number);
        // 遍历数组
        for (let i = 0; i < size; i++) {
          // arr[i]
        }
        console.log(res);
        console.log();
    }
})();
```

9. 多组测试数据，每组数据只有一个整数，对于每组输入数据，输出一行，每组数据下方有一个空行。

练习题：9. 奇怪的信

C++

```
#include<iostream>
using namespace std;
int main() {
    int n, a;
    while (cin >> n) {
        while (n != 0) {
            a = (n % 10); // 获取各位数据
            n = n / 10;
        }
        cout << result << endl;
        cout << endl;
    }
}
```

Java

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        while (in.hasNextInt()) {
            int n = in.nextInt();
            while (n > 0) {
                int tmp = n % 10; // 获取各位数据
                n /= 10;
            }
            System.out.println(res);
            System.out.println();
        }
    }
}
```

Python

```python
while 1:
    try:
        n=input()
        s=0
        for i in n:
            print(int(i))
        print(s)
        print()
    except:
        break
```

Go

```go
package main

import (
  "fmt"
)

func main() {
  var n, a int
  for {
    _, err := fmt.Scanf("%d", &n)
    if err != nil || n == 0 {
      break
    }
    for n != 0 {
      a = n % 10
      n = n / 10
    }
    fmt.Println(result)
    fmt.Println()
  }
}
```

JS

```javascript
const readline = require("readline");
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const iter = rl[Symbol.asyncIterator]();
const read_line = async () => (await iter.next()).value;

let line = null;

(async function () {
  while ((line = await read_line())) {
    const arr = line.split("").map((item) => Number(item));
    for (let i = 0; i < arr.length; i++) {
    }
    console.log(sum, "\n");
  }
})();
```

10. 多组测试数据，每个测试实例包括2个整数M，K（2<=k<=M<=1000)。M=0，K=0代表输入结束。

练习题 10. 运营商活动

C++

```
#include<iostream>
using namespace std;
int main() {
    int m, k;
    while (cin >> m >> k) {
        if (m == 0 && k == 0) break;
        int sum = m + m / k; // 第一轮回得到总话费

        cout << sum << endl;
    }
}
```

Java

```
import java.util.*;

public class Main{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (sc.hasNextInt()) {
            int m = sc.nextInt();
            int k = sc.nextInt();
            if (m == 0 && k == 0) break;
            int sum = 0;

            System.out.println(sum);
        }
    }
}
```

Python

```python
while True:
    M, K = map(int, input().split())
    if M == 0 and K == 0:
        break
    res = M
    print(res)
```

JS

```javascript
const readline = require("readline");
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const iter = rl[Symbol.asyncIterator]();

const read_line = async () => (await iter.next()).value;

let line = null;

(async function () {
  while ((line = await read_line())) {
    const [m, k] = line.split("").map((item) => Number(item));
    if (m === 0 && k === 0) {
      break;
    }
    let sum = 0;
    console.log(sum, "\n");
  }
})();
```

Go

```go
package main

import (
  "fmt"
)

func main() {
  var m, k int
  for {
    _, err := fmt.Scanf("%d %d", &m, &k)
    if err != nil || (m == 0 && k == 0) {
      break
    }
    sum := m + m/k
    fmt.Println(sum)
  }
}
```

11. 多组测试数据，首先输入一个整数N，接下来N行每行输入两个整数a和b, 读取输入数据到Map

练习题 11. 共同祖先

C++

```
#include<iostream>
#include<vector>
using namespace std;
int main() {
    int n, a, b;
    vector<int> nums = vector<int>(30, 0); // 使用数组来记录映射关系，初始化为0
    while (cin >> n) {
        while (n--) {
            cin >> a >> b;
            nums[a] = b; // 记录映射关系
        }
    }
}
```

Java

```
import java.util.*;

public class Main{
    static Map<Integer, Integer> map = new HashMap();
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (sc.hasNextInt()) {
            int n = sc.nextInt();
            for (int i = 0; i < n; i++) {
                int a = sc.nextInt();
                int b = sc.nextInt();
                map.put(a, b);
            }

        }
    }
}
```

Python

```python
while True:
    try:
        N = int(input())
        myMap = {}
        for _ in range(N):
            a, b = map(int, input().split())
            myMap[a] = b
    except:
        break
```

Js

```javascript
// 引入readline模块来读取标准输入
const readline = require('readline')

// 创建readline接口
const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

let myMap = new Array(21).fill(0)
let n
rl.on('line', (input) => {
    // 读入每行数据，将其转换为数组
    const readline = input.split(' ').map(Number)
    if (readline.length === 1) {
        n = readline[0]
    } else {
        const [a, b] = readline
        myMap[a] = b
        n--
    }
});
```

Go

```go
package main

import (
  "fmt"
)

func main() {
  var n, a, b int
  nums := make([]int, 30)

  for {
    _, err := fmt.Scanf("%d", &n)
    if err != nil {
      break
    }

    for i := 0; i < n; i++ {
      fmt.Scanf("%d %d", &a, &b)
      // 将输入数据放到map中
      nums[a] = b
    }
  }
}
```

12. 多组测试数据。每组输入一个整数n，输出特定的数字图形

练习题 12. 打印数字图形

C++

```cpp
#include<iostream>
#include<vector>
using namespace std;

void printTopPart(int n) {
    for (int i = 1; i <= n; i++) {
        // 打印空格
        for (int j = 1; j <= n - i; ++j) {
            cout << " ";
        }
        // 打印递增数字
        for (int j = 1; j <= i; j++) {
            cout << j;
        }
        // 打印递减数字
        for (int j = i - 1; j >= 1; j--) {
            cout << j;
        }
        cout << endl;
    }
}

int main() {
    int n;
    while (cin >> n) {
        if (n < 1 || n > 9) {
            cout << "输入的整数n超出范围" << endl;
        }
        printTopPart(n);
    }
}
```

Java

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        while (scanner.hasNext()) {
            int n = scanner.nextInt();
            for (int i = 1; i <= n; i++) {
                print(n - i, i);
            }
            for (int i = n - 1; i >= 1; i--) {
                print(n - i, i);
            }
        }
    }

    public static void print(int blank, int n) {
        //  前面需要补齐空格
        for (int i = 0; i < blank; i++) {
            System.out.print(" ");
        }
        for (int i = 1; i <= n; i++) {
            System.out.print(i);
        }
        for (int i = n - 1; i > 0; i--) {
            System.out.print(i);
        }
        System.out.println();
    }
}
```

Python

```python
while True:
    try:
        n = int(input())
        for i in range(1, n + 1):
            print(' ' * (n - i), end='')
            print(''.join(map(str, range(1, i + 1))) + ''.join(map(str, range(i - 1, 0,
-1))))
        for i in range(n - 1, 0, -1):
            print(' ' * (n - i), end='')
            print(''.join(map(str, range(1, i + 1))) + ''.join(map(str, range(i - 1, 0,
-1))))
    except:
        break
```

Go

```go
package main

import "fmt"

func printTopPart(n int) {
  for i := 1; i <= n; i++ {
    // 打印空格
    for j := 1; j <= n-i; j++ {
      fmt.Print(" ")
    }
    // 打印递增数字
    for j := 1; j <= i; j++ {
      fmt.Print(j)
    }
    // 打印递减数字
    for j := i - 1; j >= 1; j-- {
      fmt.Print(j)
    }
    fmt.Println()
  }
}

func main() {
  var n int
  for {
    _, err := fmt.Scan(&n)
    if err != nil {
      break
    }
		if n < 1 || n > 9 {
			fmt.Println("输入的整数n超出范围")
			continue
		}
		printTopPart(n)
	}
}
```

JS

```
function printTopPart(n) {
    for (let i = 1; i <= n; i++) {
        // 打印空格
        for (let j = 1; j <= n - i; ++j) {
            process.stdout.write(" ");
        }

        // 打印递增数字
        for (let j = 1; j <= i; j++) {
            process.stdout.write(String(j));
        }

        // 打印递减数字
        for (let j = i - 1; j >= 1; j--) {
            process.stdout.write(String(j));
        }
        console.log();
    }
}

function main() {
    const readline = require('readline');
    const rl = readline.createInterface({
        input: process.stdin,
        output: process.stdout
    });

    rl.on('line', (input) => {
        let n = parseInt(input);
        if (n < 1 || n > 9) {
            console.log("输入的整数n超出范围");
        }
        printTopPart(n);
    });
}

main();
```

13. 多行输入，每行输入为一个字符和一个整数，遇到特殊字符结束

练习题 13. 镂空三角形

C++

```
int main() {
    char c;
    int n;
    while(cin >> c){
        if(c == '@')
            break;
        cin >> n;
        myprint(c, n);
    }
    return 0;
}
```

Java

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        while (sc.hasNext()) {
            String line = sc.nextLine();
            if (line.equals("@"))
                break;

            String[] inputs = line.split(" ");
            char ch = inputs[0].charAt(0);
            int n = Integer.parseInt(inputs[1]);
        }
        sc.close();
    }

}
```

Python

```python
while True:
    try:
        line = input()
        if line == '@':
            break
        ch, n = line.split()
        n = int(n)
    except:
```

Go

```go
package main

import (
  "fmt"
)


func main() {
  var n int
  var a rune
  for {
    _, err := fmt.Scanf("%c", &a)
    if err != nil || a == '@' {
      break
    }

    _, err = fmt.Scanf("%d", &n)
    if err != nil {
      break
    }

  }
}
```

Js

```javascript
function main() {
    const readline = require('readline');
    const rl = readline.createInterface({
        input: process.stdin,
        output: process.stdout
    });

    rl.on('line', (input) => {
        let [c, n] = input.split(' ');
        if (c === '@') {
            rl.close(); // 结束输入监听
            return;
        }
        myprint(c, parseInt(n));
    });
}

main();
```

14. 第一行是一个整数n，表示一共有n组测试数据, 之后输入n行字符串

练习题 14. 句子缩写

C++

```
#include<iostream>
#include<string>
using namespace std;

int main() {
    int n;
    string result, s;
    cin >> n;
    getchar(); // 吸收一个回车，因为输入n之后，要输入一个回车
    while (n--) {
        getline(cin, s);
        for (int i = 1; i < s.size() - 1; i++) {

        }
        cout << result << endl;
    }
}
```

Java

```
import java.util.*;

public class Main{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (sc.hasNextInt()) {
            int n = sc.nextInt();
            sc.nextLine();
            for (int i = 0; i < n; i++) {
                String line = sc.nextLine().trim();

                StringBuilder sb = new StringBuilder();

                System.out.println(sb.toString());
            }
        }
    }
}
```

```
import java.util.Scanner;

public class Main{
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n=in.nextInt();
        in.nextLine();
        for (int j = 0; j < n; j++) {
            String s=in.nextLine();
            StringBuilder sb=new StringBuilder();

            System.out.println(sb.toString().toUpperCase());
        }
    }
}
```

Python

```
T = int(input())
for _ in range(T):
    words = input().split()
    for word in words:

```

Go

```
package main

import (
  "bufio"
  "fmt"
  "os"
  "strings"
)
```

```go
func main() {
  var n int
  _, err := fmt.Scan(&n)
  if err != nil {
    return
  }
  scanner := bufio.NewScanner(os.Stdin)
  for i := 0; i < n && scanner.Scan(); i++ {
    input := scanner.Text()
    words := strings.Fields(input)
    for _, word := range words {

    }
  }
}
```

Js

```js
const rl=require("readline").createInterface({input:process.stdin});
const iter=rl[Symbol.asyncIterator]();
const readline=async ()=>(await iter.next()).value;

async function main(){
  const n=parseInt(await readline());
  for(let i=0;i<n;i++){
    let line=await readline();
    let words=line.split(/\s+/);
    words.forEach(item=>{
      process.stdout.write()
    });
    console.log();
  }
}

main();
```

15. 第一行是一个整数n，然后是n组数据，每组数据2行，每行为一个字符串，为每组数据输出一个字符串，每组输出占一行

练习题 15.神秘字符

C++

```
#include<iostream>
#include<string>
using namespace std;
int main() {
    int n;
    cin >> n;
    getchar(); // 吸收n后的一个回车
    while (n--) {
        string s, t;
        cin >> s >> t;

        string result = "";

        cout << result << endl;
    }
}
```

Java

```
import java.util.Scanner;

public class Main{
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        for (int i = 0; i < n; i++) {
            String a = in.next();
            String b = in.next();
            StringBuilder sb = new StringBuilder(a);
            System.out.println(sb.toString());
        }
    }
}
```

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.StringTokenizer;

public class Main {
    public static void main(String[] args) throws IOException {
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        String str = null;
        while((str = reader.readLine())!= null){
            StringTokenizer tokenizer = new StringTokenizer(str);
            int n = Integer.parseInt(tokenizer.nextToken());
            for(int i = 0; i < n; i++){
                String a = reader.readLine();
                String b = reader.readLine();
                StringBuilder sb = new StringBuilder();

                System.out.println(sb.toString());
            }
        }
    }
}
```

Python

```
n = int(input())
for _ in range(n):
    line1 = input()
    line2 = input()
    mid = len(line1) // 2
    result = line1[:mid] + line2 + line1[mid:]
    print(result)
```

Go

```
package main

import "fmt"

func main() {
  var n int
  _, err := fmt.Scan(&n)
  if err != nil {
    return
  }
  for n > 0 {
    var a, b string
    _, _ = fmt.Scanln(&a)
    _, _ = fmt.Scanln(&b)
    fmt.Println(a[:len(a)/2] + b + a[len(a)/2:])
    n--
  }
}
```

JS

```
const rl=require("readline").createInterface({input:process.stdin});
const iter=rl[Symbol.asyncIterator]();
const readline=async ()=>(await iter.next()).value;


async function main(){
  const n=parseInt(await readline());
  for(let i=0;i<n;i++){
    let str1=await readline();
    let str2=await readline();
    console.log();
  }
}

main();
```

16. 多组测试数据，第一行是一个整数n，接下来是n组字符串，输出字符串

练习题：16. 位置互换

C++

```
#include<iostream>
#include<string>
using namespace std;

int main() {
    int n;
    cin >> n;
    string s;
    while (n--) {
        cin >> s;

        cout << s << endl;
    }
}
```

Java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.StringTokenizer;
```

```java
public class Main {
    public static void main(String[] args) throws IOException {
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        String str = null;
        while((str = reader.readLine())!= null){
            StringTokenizer tokenizer = new StringTokenizer(str);
            int n = Integer.parseInt(tokenizer.nextToken());
            for(int i = 0; i < n; i++){
                String s = reader.readLine();
                StringBuilder sb = new StringBuilder();

                System.out.println(sb.toString());
            }
        }
    }
}
```

```java
// 方法二：原地交换
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();
        sc.nextLine();
        for (int i = 0; i < n; i++) {
            String s1 = sc.nextLine();
            int len = s1.length();
            char[] chs = s1.toCharArray();

            System.out.println(new String(chs));
        }
        sc.close();
    }

}
```

Python

```
C = int(input())
for _ in range(C):
    s = input()

    print(result)
```

Go

```
package main

import (
  "fmt"
)

func main() {
  var n int

  _, err := fmt.Scanf("%d", &n)
  if err != nil {
    return
  }
  for i := 0; i < n; i++ {
    var s string
    _, err = fmt.Scanf("%s", &s)
    if err != nil {
      return
    }


    fmt.Println(s)
  }
}
```

Js

```
const rl=require("readline").createInterface({input:process.stdin});
const iter=rl[Symbol.asyncIterator]();
const readline=async ()=>(await iter.next()).value;

async function main(){
  let n=parseInt(await readline());
  for(let i=0;i<n;i++){
    let str=await readline();
    console.log();
  }
}

main();
```

17. 多组测试数据，每组测试数据的第一行为整数N（1<=N<=100），当N=0时，输入结束，第二行为N个正整数，以空格隔开，输出结果为字符串

练习题：17. 出栈合法性

C++

```
#include<iostream>
#include<stack>
#include<vector>
using namespace std;
int main() {
    int n;
    int nums[105];
    while(cin >> n) {
      // 结束输入
        if (n == 0) break;
        for (int index = 0; index < n; index++) cin >> nums[index]; // 输入数组
        stack<int> st;
        int index = 0;

      // 输出字符串
        if (st.empty() && index == n) cout << "Yes" << endl;
        else cout << "No" << endl;
    }
}
```

Java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.StringTokenizer;

public class Main {
    public static void main(String[] args) throws IOException {
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        String str = null;
        while((str = reader.readLine())!= null){
            StringTokenizer tokenizer = new StringTokenizer(str);
            // 读取n
            int n = Integer.parseInt(tokenizer.nextToken());
            if(n == 0){
                break;
            }
            int[] arr = new int[n];
            tokenizer = new StringTokenizer(reader.readLine());
          // 读取n个正整数
            for(int i = 0; i < n; i++){
                arr[i] = Integer.parseInt(tokenizer.nextToken());
            }
            if(check(arr)){
                System.out.println("Yes");
            }else{
                System.out.println("No");
            }
        }
    }
}
```

```
// 方法二：使用栈模拟
import java.util.Scanner;
import java.util.Stack;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        while (true) {
            int n = sc.nextInt();
            if (n == 0)
                break;

            Stack<Integer> stack = new Stack<>();
            for (int i = 0; i < n; i++)
```

```
                stack.push(sc.nextInt())
            }
            if (isValidPopSequence(n, poppedSequence))
                System.out.println("Yes");
            else
                System.out.println("No");
        }
        sc.close();
    }
}
```

Python

```python
while True:
    n = int(input())
    if n == 0:
        break
    sequence = list(map(int, input().split()))
    stack = []
    possible = True
    for num in sequence:
      stack.append(num)
    if possible:
        print('Yes')
    else:
        print('No')
```

Go

```go
package main

import (
  "fmt"
)

func main() {
  for {
    var n int
    _, _ = fmt.Scan(&n)

    if n == 0 {
      break
    }

    nums := make([]int, n)
    for i := 0; i < n; i++ {
      _, _ = fmt.Scan(&nums[i])
    }

    stack := make([]int, 0)

    for i := 1; i <= n; i++ {
      stack = append(stack, nums[i])
    }
  }
}
```

JS

```
const readline = require('readline')
const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
})

// 因为只能一行一行地读取数据，所以用-1来表示是否是一组新的数据
let n = -1
rl.on('line', (input) => {
    line = input.split(' ').map(Number)
    if (line.length === 1 && n === -1) {
        // 如果是一组新的数据，先读入n，直接return等待下一次输入
        n = line[0]
        return
    }
    // line保存的是出栈序列
    if (islegal(line, n)) {
        console.log('Yes')
    } else {
        console.log('No')
    }
    // 处理完一组数据后，写回n = -1
    n = -1
})
```

18. 一组输入数据，第一行为n+1个整数，逆序插入n个整数，第二行为一个整数m, 接下来有m行字符串，并根据字符串内容输入不同个数的数据

练习题：18. 链表的基本操作

C++

```
int main() {
    int n, a, m, t, z;
    string s;
    cin >> n;

    while (n--) {
        cin >> a;
    }
    cin >> m;
    while (m--) {
      // 输入m个字符串，根据字符串内容输出
        cin >> s;
        if (s == "show")  {
            cout << "Link list is empty" << endl;
        }
        if (s == "delete") {
            cin >> t;
            // 本题的删除索引是从1开始，函数实现是从0开始，所以这里是 t - 1
            if (deleteAtIndex(t - 1) == -1) cout << "delete fail" << endl;
            else cout << "delete OK" << endl;
        }
        if (s == "insert") {
            cin >> t >> z;
            if (addAtIndex(t - 1, z) == -1) cout << "insert fail" << endl;
            else cout << "insert OK" << endl;

        }
        if (s == "get") {
            cin >> t;
            int getValue = get(t - 1);
            if (getValue == -1) cout << "get fail" << endl;
            else cout << getValue << endl;
        }
    }
}
```

Java

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    // 输入n
    int n = sc.nextInt();
  // 输入n个整数
    for (int i = 0; i < n; i++) {
        int num = sc.nextInt();
        linkedList.addFirst(num);
    }
    // 输入m
    int m = sc.nextInt();

  // 输入m个字符串
    for (int i = 0; i < m; i++) {
      // 获取输入的字符串
        String operation = sc.next();
      // 根据输入内容，给出不同输出结果
        if ("get".equals(operation)) {
            int a = sc.nextInt();
            int result = linkedList.get(a - 1);
            if (result != -1) {
                System.out.println(result);
            } else {
                System.out.println("get fail");
            }
        } else if ("delete".equals(operation)) {
            int a = sc.nextInt();
            boolean deleteResult = linkedList.delete(a - 1);
            if (deleteResult) {
                System.out.println("delete OK");
            } else {
                System.out.println("delete fail");
            }
        } else if ("insert".equals(operation)) {
            int a = sc.nextInt();
            int e = sc.nextInt();
            boolean insertResult = linkedList.insert(a - 1, e);
            if (insertResult) {
                System.out.println("insert OK");
            } else {
                System.out.println("insert fail");
            }
        } else if ("show".equals(operation)) {
            linkedList.show();
        }
    }
    sc.close();
}
```

Python

```
if __name__ == "__main__":
    while True:
        mylinklist = MyLinkedList()
        try:
            # 读取链表长度和链表数值
            n, *nums = list(map(int, input().split()))
            # 初始化链表
            for i in range(n):
                mylinklist.addAtHead(nums[i])
            # 读取操作的个数
            m = int(input())
            for i in range(m):
              # 读取输入的操作和对应的索引
                s = input().split()
                if s[0] == "show":
                    if mylinklist.printLinkedList() == -1:
                        print("Link list is empty")
                if s[0] == "delete":
                    t = int(s[1])
                    if mylinklist.deleteAtIndex(t - 1) == -1:
                        print("delete fail")
                    else:
                        print("delete OK")
                if s[0] == "insert":
                    t = int(s[1])
                    z = int(s[2])
                    if mylinklist.addAtIndex(t - 1, z) == -1:
                        print("insert fail")
                    else:
                        print("insert OK")
                if s[0] == "get":
                    t = int(s[1])
                    getValue = mylinklist.get(t - 1)
                    if getValue == -1:
                        print("get fail")
                    else:
                        print(getValue)
        except:
            break
```

Go

```
func main() {
  var n int
  // 输入n
  _, err := fmt.Scan(&n)
  if err != nil {
    return
  }
  list := Constructor()

  // 输入n个数据
  for i := 0; i < n; i++ {
    var data int
    _, err = fmt.Scan(&data)
    if err != nil {
      return
    }
    list.insert(1, data)
  }
  var m int
  //输入m
  _, err = fmt.Scan(&m)
  if err != nil {
    return
  }
  // 输入m行字符串
  for i := 0; i < m; i++ {
    var s string
    _, err = fmt.Scan(&s)
    if err != nil {
      return
    }
    // 根据字符串操作输出
    switch s {
    case "get":
      var index int
      _, err = fmt.Scan(&index)
      if err != nil {
        return
      }
      val, err := list.get(index)
      if err != nil {
        fmt.Println(err.Error())
      } else {
        fmt.Println(val)
      }
    case "delete":
      var index int
      _, err = fmt.Scan(&index)
      if err != nil {
        return
      }
      err := list.delete(index)
      if err != nil {
        fmt.Println(err.Error())
      } else {
        fmt.Println("delete OK")
      }
    case "insert":
      var index, val int
      _, err = fmt.Scan(&index, &val)
```

```
        if err != nil {
          return
        }
        if err = list.insert(index, val); err != nil {
          fmt.Println(err.Error())
        } else {
          fmt.Println("insert OK")
        }
      case "show":
        list.Show()
      }
    }
  }
```

Javascript

```
const rl=require("readline").createInterface({input:process.stdin});
const iter=rl[Symbol.asyncIterator]();
const readline=async ()=>(await iter.next()).value;
const out=process.stdout;

async function main(){
  const nums=(await readline()).split(" ").map(Number);
  let root=new Node(-1);
  for(let i=1;i<=nums[0];i++){
    root.insert(1,nums[i]);
  }
  const n=parseInt(await readline());
  let index;
  for(let i=0;i<n;i++){
    let line=(await readline()).split(" ");
    let op=line[0];
    let flag=false;
    switch(op){
      case "show":
        root.show();
        break;
      case "get":
        index=parseInt(line[1]);
        let node=root.getNode(index);
        if(node) out.write(node.data.toString());
        else out.write("get fail");
        break;
      case "delete":
        index=parseInt(line[1]);
        flag=root.deleteNode(index);
        if(flag) out.write("delete OK");
```

```
          else out.write("delete fail");
          break;
        case "insert":
          index=parseInt(line[1]);
          flag=root.insert(index,parseInt(line[2]));
          if(flag) out.write("insert OK");
          else out.write("insert fail");
          break;
      }
      console.log();
    }
  }

main();
```

19. 多组测试数据，每行为n+1个数字， 输出链表或对应的字符串

练习题：19. 单链表反转

练习题：20. 删除重复元素

C++

```
int main() {

        int n, m;
        LinkedNode* dummyHead =  new LinkedNode(0); // 这里定义的头结点 是一个虚拟头结点，而不是真正的链表头结点
        while (cin >> n) {
            if (n == 0) {
                cout << "list is empty" << endl;
                continue;
            }
            LinkedNode* cur = dummyHead;
          // 读取输入构建链表
            while (n--) {
                cin >> m;
                LinkedNode* newNode = new LinkedNode(m); // 开始构造节点
                cur->next = newNode;
                cur = cur->next;
            }
            printLinkedList(dummyHead->next);
            printLinkedList(reverseList(dummyHead->next));
        }
}
// 输出链表
```

```
void printLinkedList(LinkedNode* head) {
    LinkedNode* cur = head;
    while (cur != nullptr) {
        cout << cur->val << " ";
        cur = cur->next;
    }
    cout << endl;
}
```

Java

```
import java.util.Scanner;

public class Main{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (sc.hasNextLine()) {
            String[] str = sc.nextLine().split(" ");
            if (Integer.parseInt(str[0]) == 0) {
                System.out.println("list is empty");
            }
            ListNode dummyhead = new ListNode(-1);
            ListNode cur = dummyhead;
            //构造链表
            for (int i = 1; i < str.length; i++) {
                ListNode temp = new ListNode(Integer.parseInt(str[i]));
                cur.next = temp;
                cur = cur.next;
                if (i == str.length - 1) cur.next = null;
            }
            //输出原链表
            ListNode pointer = dummyhead.next;
            while (pointer != null) {
                System.out.print(pointer.val + " ");
                pointer = pointer.next;
            }
            System.out.println();
        }
    }
}
```

python

```
# 打印链表
def printLinkedList(head: LinkedNode):
    cur = head
    while cur:
        print(cur.val, end = " ")
```

```
            cur = cur.next
        print()
if __name__ == "__main__":
        while True:
            try:
                # 输入5 1 2 3 4 5，表示链表有5个节点，值分别为1 2 3 4 5
                n, *nums = map(int, input().split())
            except:
                break
            if n == 0:
                print("list is empty")
                continue
            dummyHead = LinkedNode(0) # 这里定义的头结点 是一个虚拟头结点，而不是真正的链表头结点
            cur = dummyHead
            for i in range(n): # 开始构造节点
                cur.next = LinkedNode(nums[i])
                cur = cur.next
            printLinkedList(dummyHead.next) # 打印链表
            printLinkedList(reverseList(dummyHead.next)) # 打印翻转后的链表
```

Go

```
func main() {
  for {
    var n int
    _, err := fmt.Scan(&n)
    if err != nil {
      return
    }
    if n == 0 {
      fmt.Println("list is empty")
      continue
    }
  // 构建链表
    dummyHead := &Node{}
    cur := dummyHead
    for n > 0 {
      var val int
      _, err = fmt.Scan(&val)
      if err != nil {
        return
      }
      node := &Node{val: val}
      cur.next = node
      cur = cur.next
      n--
    }
    show(dummyHead.next)
    }
}
// 输出链表
func show(head *Node) {
    if head == nil {
        return
    }
    cur := head
    for {
        fmt.Printf("%d ", cur.val)
        cur = cur.next
        if cur == nil {
            fmt.Println()
            return
        }
    }
}
```

Js

```
// 引入readline模块来读取标准输入
const readline = require('readline')

// 创建readline接口
const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
})

// 处理输入和输出
rl.on('line', (input) => {
    // 将每一行以空格分割成一个字符串数组，并将每个元素转换成number类型
    const line = input.split(' ').filter(item => item !== '').map(Number)

    // 第一个元素是链表长度
    const n = line[0]

    // 长度为0，直接输出 list is empty
    if (n === 0) {
        console.log('list is empty')
        return
    }
    // 根据给定输入创建链表
    let head = createLinkedList(line.slice(1))
    // 打印翻转前的链表
    printLinkedList(head)
})
```

```
// 给定一个number数组，创建出链表，返回链表的头节点
function createLinkedList(arr) {
    // 创建头节点
    const head = new Node(arr[0])
    // 初始化尾指针，方便添加新的节点
    let tail = head
    arr.slice(1).forEach(item => {
        // 每次将细节点插在尾节点后面
        tail.next = new Node(item)
        // 更新尾节点为新创建的节点
        tail = tail.next
    })
    // 返回头节点
    return head
}

// 输出链表
function printLinkedList(head) {
    let output = ''
    // 将每个节点的val拼接成一个字符串
    while(head) {
        output += `${head.val} `
        head = head.next
    }
    // 最后输出
    console.log(output)
}
```

20. 多组输入，每组输入包含两个字符串，输出字符串

练习题：21. 构造二叉树

C++

```
int main() {

    string s;
    while (getline(cin, s)) { // 接受一整行字符串
        string preorder = "", inorder = "";
        // 拆分出两个字符串
        int i;
        for (i = 0; s[i] != ' '; i++) preorder += s[i];
        i++;
        for (; i < s.size(); i++) inorder += s[i];

        // 开始构造二叉树
        TreeNode* root = buildTree(preorder, inorder);
    // 输出后序遍历结果
        postorderTraversal(root);
        cout << endl;
    }
    return 0;
}

// 后序遍历二叉树
void postorderTraversal(TreeNode* root) {
    if (root == nullptr) {
        return;
    }

    postorderTraversal(root->left);
    postorderTraversal(root->right);
    cout << root->val;
}
```

Java

```
public class Main{
    public static Map<Character, Integer> map = new HashMap();
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (sc.hasNextLine()) {
            String s = sc.nextLine();
            String[] ss = s.split(" ");
            String pre = ss[0];
            String in = ss[1];
            // 构建二叉树
            TreeNode res = afterHelper(pre.toCharArray(), in.toCharArray());
            //打印二叉树
            printTree(res);
            System.out.println();
        }
    }

    public static void printTree(TreeNode root) {
        if (root == null) return;
        printTree(root.left);
        printTree(root.right);
        System.out.print(root.val);
    }
}
```

Python

```python
def postorder_traversal(root):
    if not root:
        return []
    left = postorder_traversal(root.left)
    right = postorder_traversal(root.right)
    return left + right + [root.val]

while True:
    try:
        preorder, inorder = map(str, input().split())
        if not preorder or not inorder:
            break
        root = build_tree(preorder, inorder)
        postorder = postorder_traversal(root)
        print(''.join(postorder))
    except EOFError:
        break
```

Go

```go
package main

import (
  "fmt"
  "strings"
)

func main() {
  for {
    var preorder, inorder string
    _, err := fmt.Scan(&preorder, &inorder)
    if err != nil {
      return
    }
    tree := buildTree(preorder, inorder, 0, len(preorder)-1, 0, len(inorder)-1)
    fmt.Println(postorderTraversal(tree))
  }
}
// 后序遍历结果
func postorderTraversal(root *TreeNode) string {
  var res []string
  var traversal func(node *TreeNode)
  traversal = func(node *TreeNode) {
    if node == nil {
      return
    }
```

```
        traversal(node.Left)
        traversal(node.Right)
        res = append(res, node.Val)
    }
    traversal(root)
    return strings.Join(res, "")
}
```

JS

```
const rl=require("readline").createInterface({input:process.stdin});
const iter=rl[Symbol.asyncIterator]();
const readline=async ()=>(await iter.next()).value;

const out=process.stdout;

class Node{
  constructor(data){
    this.data=data;
    this.left=null;
    this.right=null;
  }
  postOrder(){
    if(this.left!=null) this.left.postOrder();
    if(this.right!=null) this.right.postOrder();
    out.write(this.data);
  }
}

async function main(){
  while(line=await readline()){
    [preOrder,inOrder]=line.split(" ");
    const root=createBTree(preOrder,inOrder,0,preOrder.length-1,0,inOrder.length-1);
    // 获取后序遍历
    root.postOrder();
    console.log();
  }
}
```

21. 一组多行数据，第一行为数字n, 表示后面有n行，后面每行为1个字符加2个整数，输出树节点的后序遍历字符串

练习题：22. 二叉树的遍历

C++

```
#include <iostream>
#include <unordered_map>
#include <vector>
using namespace std;


// 前序遍历二叉树
void preorderTraversal(TreeNode* root) {
    if (root == nullptr) {
        return;
    }

    cout << root->val;
    preorderTraversal(root->left);
    preorderTraversal(root->right);
}

// 中序遍历二叉树
void inorderTraversal(TreeNode* root) {
    if (root == nullptr) {
        return;
    }

    inorderTraversal(root->left);
    cout << root->val;
    inorderTraversal(root->right);
}

// 后序遍历二叉树
void postorderTraversal(TreeNode* root) {
    if (root == nullptr) {
        return;
    }

    postorderTraversal(root->left);
    postorderTraversal(root->right);
    cout << root->val;
}

int main() {
    int n;
    cin >> n;
    unordered_map<char, std::pair<char, char>> nodeMap;

    // 先保存输入的数据
    vector<char> index = vector<char>(n + 1, '0');
    vector<vector<int>> nums = vector<vector<int>>(n + 1, vector<int>(2, 0));
    for (int i = 1; i <= n; i++) {
        cin >> index[i] >> nums[i][0] >> nums[i][1];
    }


    // 输出
    preorderTraversal(root);
    cout << std::endl;

    inorderTraversal(root);
    cout << std::endl;

    postorderTraversal(root);
    cout << std::endl;

    return 0;
}
```

Java

```
import java.util.*;

class TreeNode {
    char val;
    TreeNode left;
    TreeNode right;
    public TreeNode(char val) {
        this.val = val;
    }
}
public class Main{
    static TreeNode[] nodes = new TreeNode[30];

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (sc.hasNextInt()) {
            int len = sc.nextInt();
            for (int i = 0; i < len; i++) {
              // 获取字符和左右子节点
                char val = sc.next().charAt(0);
                int left = sc.nextInt();
                int right = sc.nextInt();

            }
            preorder(nodes[1]);
            System.out.println();
            inorder(nodes[1]);
            System.out.println();
            postorder(nodes[1]);
            System.out.println();
        }
    }
    public static void preorder(TreeNode root) {
        if (root == null) return;
        System.out.print(root.val);
        preorder(root.left);
        preorder(root.right);
    }
        public static void inorder(TreeNode root) {
        if (root == null) return;
        inorder(root.left);
        System.out.print(root.val);
        inorder(root.right);
    }
        public static void postorder(TreeNode root) {
        if (root == null) return;
        postorder(root.left);
        postorder(root.right);
        System.out.print(root.val);
    }
}
```

```
// 方法二：使用索引，简化构建树的过程
import java.util.Scanner;

public class Main {
    static class TreeNode {
        char val;
        int left;
        int right;

        public TreeNode(char val, int left, int right) {
            this.val = val;
            this.left = left;
            this.right = right;
        }
    }

    static TreeNode[] nodes;

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        nodes = new TreeNode[n + 1];
        for (int i = 0; i < n; i++) {
            char val = sc.next().charAt(0);
            int left = sc.nextInt();
```

```java
            int right = sc.nextInt();
            nodes[i + 1] = new TreeNode(val, left, right);
        }
        preOrderTraversal(1);
        System.out.println();
        inOrderTraversal(1);
        System.out.println();
        postOrderTraversal(1);
        System.out.println();
        sc.close();
    }

    private static void postOrderTraversal(int root) {
        if (root == 0)
            return;
        postOrderTraversal(nodes[root].left);
        postOrderTraversal(nodes[root].right);
        System.out.print(nodes[root].val);
    }

    private static void inOrderTraversal(int root) {
        if (root == 0)
            return;
        inOrderTraversal(nodes[root].left);
        System.out.print(nodes[root].val);
        inOrderTraversal(nodes[root].right);
    }

    private static void preOrderTraversal(int root) {
        if (root == 0)
            return;
        System.out.print(nodes[root].val);
        preOrderTraversal(nodes[root].left);
        preOrderTraversal(nodes[root].right);
    }

}
```

Go

```go
package main

import (
  "fmt"
  "strings"
)

func preorder(root *Node) []string {
  if root == nil {
    return []string{}
  }
  left := preorder(root.Left)
  right := preorder(root.Right)
  return append([]string{root.Val}, append(left, right...)...)
}

func inorder(root *Node) []string {
  if root == nil {
    return []string{}
  }
  left := inorder(root.Left)
  right := inorder(root.Right)
  return append(append(left, root.Val), right...)
}

func postorder(root *Node) []string {
  if root == nil {
    return []string{}
  }
  left := postorder(root.Left)
  right := postorder(root.Right)
  return append(append(left, right...), root.Val)
}

func main() {
  var n int
  fmt.Scan(&n)

  nodes := make([]*Node, n+1)
  var line string
  for i := 0; i < n; i++ {
    fmt.Scan(&line)
    val := line[0:1]
    left, right := 0, 0
    fmt.Scan(&left, &right)
  }

  root := nodes[1]
  pre := preorder(root)
  ino := inorder(root)
  post := postorder(root)

  fmt.Println(strings.Join(pre, ""))
  fmt.Println(strings.Join(ino, ""))
  fmt.Println(strings.Join(post, ""))
}
```

Python

```
def preorder(root):
    if not root:
        return []
    left = preorder(root.left)
    right = preorder(root.right)
    return [root.val] + left + right

def inorder(root):
    if not root:
        return []
    left = inorder(root.left)
    right = inorder(root.right)
    return left + [root.val] + right

def postorder(root):
    if not root:
        return []
    left = postorder(root.left)
    right = postorder(root.right)
    return left + right + [root.val]

n = int(input())
nodes = [None] * (n + 1)
line_in = []
for i in range(n):
    line = input().split()
    val, left, right = line[0], int(line[1]), int(line[2])
root = nodes[1]
pre = preorder(root)
ino = inorder(root)
post = postorder(root)
print(''.join(pre))
print(''.join(ino))
print(''.join(post))
```

JS

```
const rl=require("readline").createInterface({input:process.stdin});
const iter=rl[Symbol.asyncIterator]();
const readline=async ()=>(await iter.next()).value;

const out=process.stdout;

class Node{
  nodes=new Array();
  constructor(data,left,right){
    this.data=data;
    this.left=left;
    this.right=right;
  }
  preOrder(){
    out.write(this.data);
    if(this.left!==0) Node.nodes[this.left].preOrder();
    if(this.right!==0) Node.nodes[this.right].preOrder();
  }
  inOrder(){
    if(this.left!==0) Node.nodes[this.left].inOrder();
    out.write(this.data);
    if(this.right!==0) Node.nodes[this.right].inOrder();
  }
  postOrder(){
    if(this.left!==0) Node.nodes[this.left].postOrder();
    if(this.right!==0) Node.nodes[this.right].postOrder();
    out.write(this.data);
  }
}

async function main(){
  const n=parseInt(await readline());
  Node.nodes=new Array(n+1);
  for(let i=1;i<=n;i++){
    let line=(await readline()).split(" ");
    let left=parseInt(line[1]);
    let right=parseInt(line[2]);
    Node.nodes[i]=new Node(line[0],left,right);
  }
  Node.nodes[1].preOrder();
  console.log();
  Node.nodes[1].inOrder();
  console.log();
  Node.nodes[1].postOrder();
}

main();
```

22. 多组测试数据，首先给出正整数N，接着输入两行字符串，字符串长度为N

练习题：23. 二叉树的高度

C++

```
#include <iostream>
#include <string>
#include <unordered_map>
using namespace std;

// 计算二叉树的高度
int getHeight(TreeNode* root) {
    if (root == nullptr) {
        return 0;
    }

    int leftHeight = getHeight(root->left);
    int rightHeight = getHeight(root->right);

    return max(leftHeight, rightHeight) + 1;
}

int main() {
    int n;
    while (cin >> n) {
        string preorder, inorder;
        cin >> preorder >> inorder;

        unordered_map<char, int> indexMap;
        for (int i = 0; i < n; ++i) {
            indexMap[inorder[i]] = i;
        }
        TreeNode* root = buildTree(preorder, inorder, 0, 0, n - 1, indexMap);
        int height = getHeight(root);
        cout << height << endl;
    }
    return 0;
}
```

Java

```
// 方法一：递归
import java.util.Scanner;

public class Main {

    static class TreeNode {
        char val;
        TreeNode left;
        TreeNode right;
```

```java
        TreeNode(char val) {
            this.val = val;
            this.left = null;
            this.right = null;
        }
    }

    private static int getHeight(TreeNode root) {
        if (root == null)
            return 0;

        int leftHeight = getHeight(root.left);
        int rightHeight = getHeight(root.right);

        return Math.max(leftHeight, rightHeight) + 1;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        while (sc.hasNext()) {
            sc.nextInt();
            String preOrder = sc.next();
            String inOrder = sc.next();

            TreeNode root = buildTree(preOrder, inOrder);
            int height = getHeight(root);
            System.out.println(height);
        }

        sc.close();
    }
}
```

```java
// 方法二：递归（使用哈希表来优化中序遍历中查找根节点位置的过程）
import java.util.HashMap;
import java.util.Scanner;

public class Main {

    static class TreeNode {
        char val;
        TreeNode left;
        TreeNode right;

        TreeNode(char val) {
            this.val = val;
            this.left = null;
            this.right = null;
        }
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        while (sc.hasNext()) {
            int N = sc.nextInt();
            String preOrder = sc.next();
            String inOrder = sc.next();

            HashMap<Character, Integer> inOrderMap = new HashMap<>();
            for (int i = 0; i < N; i++) {
                inOrderMap.put(inOrder.charAt(i), i);
            }

            TreeNode root = buildTree(preOrder, 0, N - 1, 0, N - 1, inOrderMap);
            int height = getHeight(root);
            System.out.println(height);
        }

        sc.close();
    }

    private static int getHeight(TreeNode root) {
        if (root == null) {
            return 0;
        }

        int leftHeight = getHeight(root.left);
        int rightHeight = getHeight(root.right);

        return Math.max(leftHeight, rightHeight) + 1;
    }
}
```

Python

```
class TreeNode:
    def __init__(self, val):
        self.val = val
        self.left = None
        self.right = None

def get_height(root):
    if not root:
        return 0

    left_height = get_height(root.left)
    right_height = get_height(root.right)

    return max(left_height, right_height) + 1

def main():
    while True:
        try:
            N = int(input())
            pre_order = input().strip()
            in_order = input().strip()

            in_order_map = {}
            for i in range(N):
                in_order_map[in_order[i]] = i
            root = build_tree(pre_order, 0, N - 1, 0, N - 1, in_order_map)
            height = get_height(root)
            print(height)
        except EOFError:
            break

if __name__ == "__main__":
    main()
```

Go

```
package main

import "fmt"

// 定义二叉树结构体
type treeNode struct {
    val   byte      // 节点的值
    left  *treeNode // 左子树
    right *treeNode // 右子树
}

// 计算二叉树的高度（深度）
func height(root *treeNode) int {
    if root == nil {
        return 0
    }

    leftHeight := height(root.left)
    rightHeight := height(root.right)
    if leftHeight > rightHeight {
        return leftHeight + 1
    } else {
        return rightHeight + 1
    }
}

func main() {
    var k int
    for {
        _, err := fmt.Scan(&k)
        if err != nil {
            break
        }
        preorder := make([]byte, k)
        inorder := make([]byte, k)

        fmt.Scan(&preorder, &inorder)

        // 构建二叉树
        root := buildTree(preorder, inorder)

        // 计算二叉树高度
        fmt.Println(height(root))
    }
}
```

JS

```
const rl=require("readline").createInterface({input:process.stdin});
const iter=rl[Symbol.asyncIterator]();
const readline=async ()=>(await iter.next()).value;

const out=process.stdout;

class Node{
  constructor(data){
    this.data=data;
    this.left=null;
    this.right=null;
  }
  getHeight(){
    let leftHeight=0,rightHeight=0;
    if(this.left!=null) leftHeight=this.left.getHeight();
    if(this.right!=null) rightHeight=this.right.getHeight();
    return Math.max(leftHeight,rightHeight)+1;
  }
}
```

```
async function main(){
  while(line=await readline()){
    // 获取输入的n
    n=parseInt(line);
    // 获取第一行字符串
    let preOrder=await readline();
    // 获取第二行字符串
    let inOrder=await readline();
    let root=Node.createTree(preOrder,inOrder,0,n-1,0,n-1);
    console.log(root.getHeight());
  }
}

main();
```

23. 多组测试数据。每组输入占一行，为两个字符串，由若干个空格分隔

练习题：24.最长公共子序列

C++

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;
int main() {
    string text1, text2;
    while (cin >> text1 >> text2) {
      // 初始化dp数组
        vector<vector<int>> dp(text1.size() + 1, vector<int>(text2.size() + 1, 0));

      // 输出结果
        cout << dp[text1.size()][text2.size()] << endl;
    }
    return 0;
}
```

Java

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        while (scanner.hasNextLine()) {
            String line = scanner.nextLine();
            String[] s = line.split(" ");
            String x = s[0];
            String y = s[1];
            int m = x.length();
            int n = y.length();
            // 初始化dp数组
            int[][] dp = new int[m + 1][n + 1];
            // 输出
            int max = dp[m][n];
            System.out.println(max);
        }
    }

}
```

Go

```
package main

import (
  "fmt"
)


func longestCommonSubsequence(X, Y string) int {
  m := len(X)
  n := len(Y)
  // 创建一个二维数组dp
  dp := make([][]int, m+1)
  for i := 0; i <= m; i++ {
    dp[i] = make([]int, n+1)
  }

  return dp[m][n]
}

func main() {
  var X, Y string
  for {
    // 输入两个字符串
    _, err := fmt.Scan(&X, &Y)
    if err != nil {
      break
    }
    result := longestCommonSubsequence(X, Y)
    fmt.Println(result)
  }
}
```

Python

```
while True:
    try:
        text1, text2 = input().split()
    except:
        break

    dp = [[0] * (len(text2) + 1) for _ in range(len(text1) + 1)]

    print(dp[len(text1)][len(text2)])
```

Js

```
const readline = require('readline');

const rl = readline.createInterface({
    input: process.stdin,
    output:process.stdout,
})

rl.on('line',function(line){
    const input = line.split(' ');
    const str1 = input[0], str2 = input[1];
    const len1 = str1.length, len2 = str2.length;
  // 初始化dp数组
    const dp = new Array(len1 + 1).fill(0).map(() => new Array(len2 + 1).fill(0));

  // 输出
    console.log(dp[len1][len2]);
})
```

24. 多组测试数据，每组第一行为两个正整数n和m，接下来m行，每行3个整数, 最后一行两个整数

练习题：25. 最爱的城市

C++

```
int main() {
    int n, m;
    while (cin >> n >> m) {
        // 构建图
        while (m--) {
            int a, b, l;
            std::cin >> a >> b >> l;
        }

        int x, y;
        std::cin >> x >> y;

    }
    return 0;
}
```

```
int main() {
    int n, m;
    while (cin >> n >> m) {
        // 构建图
        for (int i = 0; i < m; i++) {
            int a, b, l;
            cin >> a >> b >> l;
        }

        int x, y;
        std::cin >> x >> y;

    }
    return 0;
}
```

Java

```
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        while (scanner.hasNext()) {
          //  处理输入
            int n = scanner.nextInt();
            int m = scanner.nextInt();

            for (int i = 0; i < m; i++) {
                int a = scanner.nextInt();
                int b = scanner.nextInt();
                int l = scanner.nextInt();
            }
            int x = scanner.nextInt();
            int y = scanner.nextInt();

            // 处理输出
            int res = dfs(graph, x, y, isVisit, sum);
            if (res != Integer.MAX_VALUE) {
                System.out.println(res);
            } else {
                System.out.println("No path");
            }
        }
    }

    private static int dfs(int[][] graph, int start, int end, int[] isVisit, int sum) {
        if (end == start) {
            return sum;
        }
        return min;
    }
}
```

Python

```python
def main():
    while True:
        try:
          # 接收一行作为输入，将之分隔成n, m
            n, m = map(int, input().split())
            # 接收m行作为输入
            for i in range(m):
                a, b, l = map(int, input().split())
            # 接收一行作为输入，将之分隔成x, y
            x, y = map(int, input().split())
            if dist[x][y] == float('inf'):
                print("No path")
            else:
                print(dist[x][y])

        except EOFError:
            break


if __name__ == "__main__":
    main()
```

Go

```go
func main() {
  for {
    // 接收n和m
    var n, m int
    if _, err := fmt.Scan(&n, &m); err != nil {
        break
      }
    // 接收m行数据
    for i := 0; i < m; i++ {
      var a, b, l int
      fmt.Scan(&a, &b, &l)
    }

    // 接收最后一行两个数据
    var x, y int
    fmt.Scan(&x, &y)

    if graph[x][y] != math.MaxInt32 {
      fmt.Println(graph[x][y])
    } else {
      fmt.Println("No path")
    }
  }
}
```

JS

```
const readline = require('readline')
const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
})
let m, n
let input = []
rl.on('line', (line) => {
    input.push(line)
}).on('close', () => {
    let index = 0
    while(index < input.length) {
        const [n,m] = input[index++].split(' ').map(Number)
        const edges = []
        for(let i = 0; i < m; i++) {
            const [a,b,l] = input[index++].split(' ').map(Number)
            edges.push([a,b,l])
        }
        const [x,y] = input[index++].split(' ').map(Number)
        const result = floydWarshall(n,m,edges,x,y)
        if(result === INF) {
            console.log('No path')
        } else {
            console.log(result)
        }
    }
})
```