模拟练习试题参考答案（Java）

为了帮助大家熟悉 CCF 软件能力认证考试的操作方式与答题环境，了解试题的大致难度，做好考前的准备，故在此提供试题的参考答案。Java 程序是灵活的，为了解决同一个问题，即使结果相同，程序的内容也不一定是完全一致的，仅供各位在练习时参考。

1. 出现次数最多的数

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        new Main().run();
    }

    public void run() {
        Scanner fin = new Scanner(System.in);

        int N = fin.nextInt();
        int[] count = new int[10001];
        for (int i = 0; i < N; ++i) {
            ++count[fin.nextInt()];
        }

        int maxCount = -1;
        int result = 0;
        for (int i = 1; i <= 10000; ++i) {
            if (count[i] > maxCount) {
                maxCount = count[i];
                result = i;
            }
        }

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

2. ISBN 号码

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
```

```
public class Main {
	public static void main(String args[]) {

		BufferedReader bin = new BufferedReader(new InputStreamReader(System.in));
		try{
			int sum=0;char cc='0';
			String isbn_0 = bin.readLine();
			String isbn = isbn_0.replace("-", "");

			for(int i=0; i<9; i++){
				int ii = (int)isbn.charAt(i)-48;
				sum += ii * (i+1);
			}
			sum = sum % 11;
			if(sum == 10) cc = 'X';//
			else cc = (char)(sum+48);
			if(cc == isbn.charAt(9)) System.out.println("Right");
			else{
				isbn_0 = isbn_0.substring(0,12) + cc;
				System.out.println(isbn_0);
			}
		}catch(Exception e){
			e.printStackTrace();
		}
	}
}
```

3. 最大的矩形

```
import java.util.*;

public class Main {
   public static void main(String[] args) {
      new Main().run();
   }

   public void run() {
      Scanner fin = new Scanner(System.in);
```

```
        int N = fin.nextInt();
        int[] height = new int[N];
        for (int i = 0; i < N; ++i) height[i] = fin.nextInt();

        int result = 0;
        for (int i = 0; i < N; ++i) {
            int width = 1;
            for (int j = i - 1; j >= 0; --j) {
                if (height[j] < height[i]) break;
                ++width;
            }
            for (int j = i + 1; j < N; ++j) {
                if (height[j] < height[i]) break;
                ++width;
            }
            int area = width * height[i];
            result = Math.max(result, area);
        }

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

4. 有趣的数

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        new Main().run();
    }

    public void run() {
        Scanner fin = new Scanner(System.in);

        int N = fin.nextInt();
        long[] count = new long[8];
        count[6] = 0;
        count[7] = 1;
        long mod = 1000000007;
```

```
        for (int i = 2; i <= N; ++i) {
            long[] newCount = new long[8];
            newCount[0] = (count[0] * 2 + count[1] + count[3]) % mod;
            newCount[1] = (count[1] * 2 + count[2] + count[5]) % mod;
            newCount[2] = (count[2] + count[6]) % mod;
            newCount[3] = (count[3] * 2 + count[4] + count[5]) % mod;
            newCount[4] = (count[4] + count[7]) % mod;
            newCount[5] = (count[5] * 2 + count[6] + count[7]) % mod;
            newCount[6] = 0;
            newCount[7] = 1;

            count = newCount;
        }

        System.out.println(count[0]);
    }
}
```

5. I’m stuck!

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        new Main().run();
    }

    public void run() {
        Scanner fin = new Scanner(System.in);

        int R = fin.nextInt();
        int C = fin.nextInt();
        fin.nextLine();
        int[][] board = new int[R + 2][C + 2];
        int rowStart = 0, colStart = 0, rowEnd = 0, colEnd = 0;
        for (int i = 1; i <= R; ++i) {
            String line = fin.nextLine();
            for (int j = 1; j <= C; ++j) {
                char c = line.charAt(j - 1);
                switch (c) {
```

```
            case '#': break;
            case '-': board[i][j] = 5; break;
            case '|': board[i][j] = 0xA; break;
            case '+':
            case 'S':
            case 'T':
                        board[i][j] = 0xF; break;
            case '.': board[i][j] = 0x8; break;
            default: break;
        }

        if (c == 'S') {
            rowStart = i;
            colStart = j;
        } else if (c == 'T') {
            rowEnd = i;
            colEnd = j;
        }
    }
}

int[] dr = new int[] {0, -1, 0, 1};
int[] dc = new int[] {1, 0, -1, 0};

// Scan 1: find all cells which can reach T
boolean[][] visited = new boolean[R + 2][C + 2];
boolean[][] canReachT = new boolean[R + 2][C + 2];
initVisited(visited);
canReachT[rowEnd][colEnd] = true;
visited[rowEnd][colEnd] = true;

Queue<Integer> queue = new LinkedList<Integer>();
queue.add(rowEnd);
queue.add(colEnd);
while (!queue.isEmpty()) {
    int r = queue.remove();
    int c = queue.remove();

    for (int i = 0; i < 4; ++i) {
        int nr = r + dr[i];
```

```
        int nc = c + dc[i];
        if (visited[nr][nc]) continue;
        if ((board[nr][nc] & (1 << ((i + 2) % 4))) != 0) {
            canReachT[nr][nc] = true;
            queue.add(nr);
            queue.add(nc);
            visited[nr][nc] = true;
        }
    }
}

/*
for (int i = 1; i <= R; ++i) {
    for (int j = 1; j <= C; ++j) {
        if (canReachT[i][j]) {
            System.out.println("i = " + i + ", j = " + j);
        }
    }
}
*/
if (!canReachT[rowStart][colStart]) {
    System.out.println("I'm stuck!");
    return;
}

// Scan 2: get result
boolean[][] rCanReach = new boolean[R + 2][C + 2];
initVisited(visited);
queue.clear();
visited[rowStart][colStart] = true;
rCanReach[rowStart][colStart] = true;
queue.add(rowStart);
queue.add(colStart);

while (!queue.isEmpty()) {
    int r = queue.remove();
    int c = queue.remove();

    for (int i = 0; i < 4; ++i) {
        if ((board[r][c] & (1 << i)) == 0) continue;
```

```
            int nr = r + dr[i];
            int nc = c + dc[i];
            if (visited[nr][nc]) continue;
            if (board[nr][nc] == 0) continue;

            rCanReach[nr][nc] = true;
            queue.add(nr);
            queue.add(nc);
            visited[nr][nc] = true;
        }
    }

    int result = 0;
    for (int i = 1; i <= R; ++i) {
        for (int j = 1; j <= C; ++j) {
            /*
            if (rCanReach[i][j]) {
                System.out.println("i = " + i + ", j = " + j);
            }
            */
            if (rCanReach[i][j] && (!canReachT[i][j])) ++result;
        }
    }
    System.out.println(result);
}

private void initVisited(boolean[][] visited) {
    int R = visited.length - 2;
    int C = visited[0].length - 2;
    for (int i = 0; i <= R + 1; ++i) {
        visited[i][0] = true;
        visited[i][C + 1] = true;
    }
    for (int j = 0; j <= C + 1; ++j) {
        visited[0][j] = true;
        visited[R + 1][j] = true;
    }
    for (int i = 1; i <= R; ++i) {
        for (int j = 1; j <= C; ++j) {
            visited[i][j] = false;
```

```
        }
      }
    }
  }
```