论文题目：B

组　　别：**本科生**

参赛队员信息：

	姓名	学部（学院）	学号	联系电话
队员 1	巫潼钰	化工海洋与生命学院	20243121819	18848321851
队员 2	唐菀婷	化工海洋与生命学院	20243121821	13477778900
队员 3	李民	令希书院	20251055150	14577778900

参赛学校：

大连理工大学

大型展销会临时工招聘与排班优化问题

摘 要

关键词:

一、问题重述

1.1 问题背景

大型展销会通常具有参展企业多、观众流量大、服务环节复杂等特点，对现场组织与人力调度提出了较高要求。为确保展销会在为期 10 天的会期内高效、有序地运行，主办方需根据不同工作小组的职责安排大量临时工参与现场服务。由于各小组在不同日期、不同小时的人力需求存在显著差异，人力部门必须在会前统一完成临时工的招聘与排班设计，使得在满足所有岗位需求的同时，将总体用工数量控制在尽可能低的水平。

临时工的工作制度具有明确的约束：每名临时工每天工作 8 小时，由两个连续的 4 小时工作段组成；在整个 10 天会期中，每名临时工必须安排 2 天休息，即实际工作 8 天。此外，10 个小组每天的工作时段均为 8:00 至 19:00（共 11 小时），并已提前上报了每小时的用工需求。如何在严格的时间结构、休息制度与岗位需求约束下，合理安排临时工的工作与休息，是一个典型的排班优化问题。

在实际管理中，不同的用工模式会显著影响排班的灵活性与所需临时工数量。例如，临时工是否可以跨天更换小组、是否可以在同一天服务多个小组等，都将改变排班的可行性与优化空间。因此，有必要在不同用工模式下分别建立数学模型，求解最少所需临时工数量，并给出可实施的排班方案，为展销会人力资源管理提供科学依据。

1.2 具体重述

本题给出了 10 个小组在 10 天中每天每小时的临时工需求量，并规定了临时工的工作制度与休息要求。需要在人力部门统一招聘的前提下，为临时工安排工作与休息，使得所有小组在所有时段的需求均得到满足，同时使招聘人数最少。

根据不同的用工模式，需要分别建立数学模型并求解：

1.2.1 问题 1：固定小组模式

每名临时工在整个 10 天中只能服务于同一个小组。要求在此约束下，确定最少需要招聘的临时工数量，并给出具体排班方案。

1.2.2 问题 2：跨天换组模式

每名临时工在同一天内只能服务于一个小组，但不同天可以更换小组。需要在此模式下重新建立模型，求解最少临时工数量与排班方案。

1.2.3 问题 3：同日跨组模式

每名临时工不仅可以跨天更换小组，还可以在同一天内服务至多两个小组。要求每个小组的服务时段必须是连续 4 小时，且两个工作时段之间至少休息 2 小时。在此更灵活的模式下，求解最少临时工数量与排班方案。

二、问题分析

本题给出了大型展销会 10 个小组在 10 天中每天每小时的临时工需求量。每名临时工每天必须工作两个连续的 4 小时时段（共 8 小时），并在整个 10 天中休息恰好 2 天，即实际工作 8 天。目标是在满足所有小组、所有日期、所有小时的需求约束下，使所需临时工总人数最少。

为系统刻画临时工的排班结构，可将排班方案分解为“天层次—小时层次—小组层次”三个维度，每名临时工的完整排班由以下三部分唯一确定：

（1）天层次：休息日选择

临时工必须在 10 天中休息恰好 2 天，因此共有

$$C_{10}^{2} = 45$$

种不同的休息日组合。记第 k 种休息方案为 $k = 1, 2, \ldots, 45$。每个方案明确规定了该临时工在 10 天中哪些天工作、哪些天休息。

（2）小时层次：每日工作时段选择

每天必须选择两个不重叠的连续 4 小时时段。根据 8:00–19:00 的时间范围，可枚举所有合法的时段组合，例如：

- 8:00–12:00 与 12:00–16:00；
- 9:00–13:00 与 15:00–19:00；
- 10:00–14:00 与 14:00–18:00；
- 11:00–15:00 与 15:00–19:00；

设所有合法组合共有 M 种，编号为 $m = 1, 2, \ldots, M$。每个组合对应一组覆盖小时集合 $\mathcal{H}_m$。

（3）小组层次：所属小组选择

在问题 1 中，每名临时工在 10 天内只能服务于同一个小组，因此需指定其所属小组

$$g = 1, 2, \ldots, 10.$$

（4）排班方案变量

基于上述三层结构，每名临时工的排班方案可表示为三元组 (k, m, g)。设

$$X_{k,m,g}$$

表示采用休息方案 k、每日采用时段组合 m、并服务于小组 g 的临时工人数，则所有临时工的排班均可由这些变量线性组合得到。

（5）需求满足约束

对任意小组 g、任意日期 d、任意小时 h，所有在该时段工作的临时工人数必须满足需求量 $R_{g,d,h}$。由于休息方案 k 决定某工人在第 d 天是否上班，时段组合 m 决定其覆盖的小时集合 $\mathcal{H}_m$，因此需求约束可写为：

$$\sum_{k=1}^{45}\sum_{m=1}^{M} X_{k,m,g}\cdot \mathbf{1}_{(d\text{ 为方案}k\text{ 的工作日})}\cdot \mathbf{1}_{(h\in\mathcal{H}_m)} \;\geq\; R_{g,d,h},\quad \forall g,d,h.$$

（6）休息日与工作时段合法性

每个休息方案 k 已满足“休息 2 天、工作 8 天”的要求；每个时段组合 m 已满足“两个连续 4 小时段、不重叠”的要求，因此无需额外约束。

（7）非负整数约束

$$X_{k,m,g}\in\mathbb{Z}_{\geq 0}.$$

（8）目标函数

目标是使所需临时工总人数最少，即：

$$\min \sum_{k=1}^{45}\sum_{m=1}^{M}\sum_{g=1}^{10} X_{k,m,g}.$$

（9）小结

通过将排班方案拆解为“休息日选择—工作时段选择—服务小组选择”三层结构，可将复杂的排班问题转化为一个具有明确组合结构的整数规划模型。变量 $X_{k,m,g}$ 对应一类完全确定的排班模式，模型通过需求约束确保所有小时的用工需求得到满足，通过目标函数实现临时工人数最小化。该建模框架结构清晰、可扩展性强，并可自然推广至问题 2 与问题 3。

三、问题假设

本研究提出如下假设：

1.

四、符号说明

符号	解释	单位
符号 1	解释 1	单位 1
符号 2	解释 2	
符号 1	解释 1	单位 3

五、模型的建立与求解

5.1 对问题 1 的求解

六、灵敏度分析

七、模型评价

7.1 模型优点

7.2 模型不足

八、模型的改进与推广

8.1 模型改进

8.2 模型推广

参 考 文 献

附录 1　相关定理的证明

附录 2　程序代码

基础程序

```
    int main(){
        int i;
        printf("hello latex!\n");
        return 0;
    }
```

计算 n 的阶乘（C++）：

```
#include<iostream>

long long factorial(int n){
    if (n <= 1) {
        return 1;
    }
    return n * factorial(n - 1);
}

int main(){
    int n;
    std::cout <<"请输入一个整数：";
    std::cin >> n;
    if (n < 0) {
        std::cout <<"请输入一个非负整数。" << std::endl;
        return 1;
    }
    std::cout << n <<"的阶乘是：" << factorial(n) << std::endl;
    return 0;
}
```

计算 n 的阶乘（Java）：

```
public class FactorialCalculator {

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

        System.out.print("请输入一个非负整数：");
        int n = scanner.nextInt();
```

```
        if (n < 0) {
            System.out.println("输入错误！请输入一个非负整数。");
        } else {
            long factorial = calculateFactorial(n);
            System.out.println(n + "的阶乘是: " + factorial);
        }

        scanner.close();
    }

    public static long calculateFactorial(int n) {
        long result = 1;
        for (int i = 1; i <= n; i++) {
            result *= i;
        }
        return result;
    }
}
```

计算 n 的阶乘（Python）：

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n-1)

# 测试代码
n = int(input("请输入一个非负整数: "))
if n < 0:
    print("输入错误！请输入一个非负整数。")
else:
    result = factorial(n)
    print(f"{n}的阶乘是: {result}")
```

计算 n 的阶乘（MATLAB）：

```
function result = factorial(n)
    if n < 0
        error('输入错误！请输入一个非负整数。');
    elseif n == 0 || n == 1
        result = 1;
    else
```

```
        result = 1;
        for i = 2:n
            result = result * i;
        end
    end
end

% 测试代码
n = input('请输入一个非负整数: ');
try
    result = factorial(n);
    fprintf('%d的阶乘是: %d\n', n, result);
catch ME
    if strcmp(ME.identifier, 'MATLAB:narginchk:notEnoughInputs
       ')
        fprintf('没有输入任何数字。\n');
    else
        fprintf('发生错误: %s\n', ME.message);
    end
end
```