

2024 年“中银杯”甘肃省职业院校技能大赛 高职学生组电子与信息大类 区块链技术应用赛项竞赛规程

一、赛项名称

赛项名称：区块链技术应用

赛项组别：高职学生组

竞赛形式：团体赛

赛项归属产业：电子信息大类

二、竞赛目的

区块链技术应用赛项围绕区块链技术在产业应用中的工作岗位技能要求而设计，赛项内容覆盖区块链产业主流的技术方向。通过大赛培养参赛选手在企业真实项目环境下进行区块链平台框架搭建、区块链产品需求分析与方案设计、区块链系统部署、区块链系统运维与监测、智能合约开发、区块链应用软件前端与后端开发、区块链系统测试及调优等方面的能力，形成良好的职业素养，全面提升学生自主解决综合问题的能力，达到“以赛促教、以赛促学、以赛促改、赛课融通、赛训结合”目的。

通过赛项的设置，加强职业院校与区块链产业的衔接，引导院校与企业共同开发区块链课程和资源，促进教师开展区块链关键应用技术研究，推进区块链技术技能人才培养，实现产业链、创新链与教育链协同创新，促进产教融合和科

教融汇，服务国家“网络强国、数字中国”战略。

三、竞赛时间、地点

竞赛时间：全省职业院校技能大赛执委会办公室统一通知

竞赛地点：兰州职业技术学院总校区

竞赛日程：如下表 1

表 1-竞赛日程表

日期	时间	内容	参加人员	地点
竞赛前 2 日	18:00 前	裁判、专家、监督员报到	裁判、专家组、监督员	住宿酒店
竞赛前 1 日	09:00-14:00	参赛队报到,安排住宿, 领取资料	工作人员、参赛队	住宿酒店
	09:00-12:00	裁判培训会议	裁判、专家组、监督	会议室
	13:00-14:30	裁判工作会议	裁判、监督员	会议室
	15:00-15:30	开幕式	领导、各参赛队、裁判长、专家组、监督员	会议室
	15:30-16:00	领队会,抽签确定检录顺序号	各参赛队领队、裁判长	会议室
	16:00-16:40	熟悉赛场	各参赛队	竞赛场地
	16:40	检查封闭赛场	裁判长、监督员	竞赛场地
	17:00	参赛队返回酒店	各参赛队	
竞赛日	07:30	参赛队到达竞赛场地前集合	各参赛队、工作人员	竞赛场地
	07:30-08:45	1) 大赛检录 2) 工位号抽签	1) 参赛选手、检录裁判、监督员 2) 参赛选手、加密裁判、监督员	签到区域
	08:45-09:00	环境确认、赛题发放	参赛选手、现场裁判、监督员	竞赛场地
	09:00-13:00	实操比赛	参赛选手、现场裁判、监督员	竞赛场地
	14:00-18:30	现场评分、抽检、统分、复验	裁判、监督员	竞赛场地

四、竞赛内容

（一）选手需具备的能力

区块链技术应用赛项根据《全国职业院校技能大赛执行规划（2023—2027 年）》，结合高职区块链技术应用专业简介，针对区块链新兴数字产业所需的技术技能，面向区块链应用开发、智能合约开发、区块链测试、区块链运维、区块链运营等岗位，区块链应用设计与开发、区块链平台部署与运维、智能合约开发与测试、区块链应用软件开发等典型工作任务，基于企业实际项目，要求选手在规定时间内完成指定任务的区块链应用开发。赛项主要考查选手对区块链系统应用需求分析与方案设计，区块链应用及智能合约设计与开发，区块链系统测试设计、执行与分析，区块链系统部署、维护和监控，基于区块链系统的应用软件前端与后端开发等专业核心能力及职业素养，全面检验学生在区块链技术应用的工程实践能力和创新能力，展现高职区块链人才培养成果。竞赛技能要求及成绩比例见表 2。

表 2 竞赛技能要求及成绩比例

竞赛内容	技能要求	成绩比例
区块链系统部署与运维	1. 熟悉 Linux 操作系统运维 2. 熟练使用 Docker 容器 3. 熟悉常见的区块链技术架构及运行机制，能搭建和配置区块链平台及网络 4. 熟练使用 MySQL 等主流数据库，完成业务系统数据库的创建和管理 5. 熟练使用脚本、编程语言和日志分析工具快速定位问题 6. 熟悉区块链访问接口、数据格式，能快速分析区块	25%

竞赛内容	技能要求	成绩比例
	链节点状态和系统运行状态	
区块链系统测试	1. 具备测试需求分析及测试用例设计能力 2. 熟练使用常见的测试工具 3. 能对区块链系统进行防篡改测试和签名测试 4. 具备全链路压力测试设计和执行能力	10%
智能合约开发	1. 熟悉 Solidity 基本语法 2. 具备智能合约编程的能力 3. 具备智能合约部署和调用的能力	20%
智能合约测试	1. 熟悉智能合约的运行机制 2. 具备智能合约测试的能力	10%
区块链应用 前端开发	1. 熟悉常用前端开发技术 2. 具备 UI 设计能力 3. 具备页面逻辑编程能力	10%
区块链应用 后端开发	1. 熟悉 Java 等后端开发语言及框架技术 2. 具备后端编程调用智能合约能力 3. 具备后端编程访问数据库能力	20%
职业素养	1. 具有良好的文档写作能力，代码编写规范 2. 具有团队合作精神和创新意识	5%

（二）竞赛模块

表 3 竞赛模块内容及时长

模块	主要内容	比赛时长	分值
模块一	区块链网络部署 根据项目背景描述完成区块链产品的需求分析与方案设计，在 Linux 环境下完成区块链系统的部署、运维及测试	90min	35 分
模块二	智能合约开发与测试 使用 Solidity 编程语言开发智能合约，设计符合需求的合约接口，完成合约功能的开发，对智能合约进行编译、部署和调用；进行智能合约测试	75min	30 分
模块三	区块链应用系统开发 利用前端开发语言及框架完成页面逻辑设计和展示；利用 Java 等后端	75min	30 分

模块		主要内容	比赛 时长	分值
		开发语言及框架，实现应用程序接口，完善区块链应用系统，调用智能合约实现链上信息的查询和结果展示		
职业素养		团队分工合理、操作规范、文明竞赛		5 分
合计			240min	100 分

1. 模块一：区块链网络部署

(1) 基于给定的环境和区块链系统，完成系统部署及节点部署。使用监控工具完成对网络、节点服务的监控。根据业务需求，完成系统日志、网络参数、节点服务等系统结构的维护。

(2) 设计区块链系统的测试流程，调用智能合约进行单元测试、集成测试、系统测试和性能测试；根据业务需求，分析并修复给定智能合约中的安全漏洞。

2. 模块二：智能合约开发与测试

(1) 使用 Solidity 语言进行智能合约开发，完成智能合约的部署和调用。

(2) 编写智能合约单元测试代码并完成智能合约的功能测试和性能测试。

3. 模块三：区块链应用系统开发

(1) 根据业务需求，使用前端开发框架完成页面设计，使用已提供的服务端接口获取业务数据，并进行部署与展示。

(2) 依据功能需求,使用 Java 等后端开发语言及常用框架进行后端代码开发,访问数据库、实现应用程序接口、调用智能合约,完善区块链应用系统,完成后端代码的部署。

4. 职业素养

要求参赛选手文档写作科学规范,具有团队合作精神和创新意识,比赛操作严谨,代码编写规范,文明竞赛。

(三) 任务书库(样例)

模块一:区块链产品方案设计及系统运维(35分)

任务 1-1:区块链系统部署与运维(25分)

围绕供应链金融区块链平台部署与运维需求,进行项目相关系统、节点以及管理工具的部署工作。通过监控工具完成对网络、节点服务的监控。最终根据业务需求规范,完成系统日志、网络参数、节点服务等系统结构的维护。

1. 根据参数与端口设置要求,部署区块链系统并验证;
2. 根据参数与端口设置要求,部署区块链网络管理平台并验证;
3. 基于区块链系统相关管理平台,按照任务指南实施系统运维工作并验证。
4. 基于区块链系统相关监管工具,按照任务指南对区块链系统进行监管。

子任务 1-1-1:搭建区块链系统并验证(8分)

基于给定服务器环境以及软件(地址“/root/tools”),使用 Docker 以默认配置安装单机 4 节点的区块链系统,并完成控制台工具的部署:

- (1) 完成系统搭建配置与启动。(2分)
- (2) 使用基于 Docker 命令查看区块链系统状态。(2分)
- (3) 检查区块链系统节点 node0 连接状态输出。(2分)
- (4) 配置控制台,管理相关证书并启动。(2分)

子任务 1-1-2:区块链管理平台部署与验证(8分)

基于给定服务器环境以及软件(地址“/root/tools”),按要求部署区块链管理平台,具体工作如下:

- (1) 配置 Mysql 数据库(2分)

(2) 配置管理平台连接区块链系统（2分）

(3) 使用命令启动管理平台服务（2分）

(4) 验证管理平台启动情况（2分）

子任务 1-1-3：区块链系统节点运维（5分）

基于已完成的区块链系统与管理平台搭建工作，开展相关节点运维工作：

(1) 生成新节点(node4)，启动并检查（2分）

(2) 修改新节点配置，并查看节点的 nodeid（2分）

(3) 将新节点作为观察节点加入 group1 当中，并检查是否加入成功（1分）

子任务 1-1-4：区块链系统管理平台运维（4分）

基于已部署的区块链系统管理平台，进行系统相关运维工作：

(1) 基于管理平台功能页面，添加新主机（2分）

(2) 基于管理平台功能页面，修改新节点（node4）节点状态，并监控。（2分）

任务 1-2：区块链系统测试（10分）

设计对区块链系统的测试流程；结合实际业务需求，调用部署的智能合约中进行系统测试、性能测试等；根据业务需求，分析并且修复给定智能合约中的安全漏洞。利用模拟业务和测试工具来完成对区块链系统服务数据的测试。

1. 基于区块链系统的中间件服务的部署脚本完成中间件服务环境搭建以及搭建结果验证，最后将执行结果截图保存。（3分）

(1) 实现区块链系统中间件服务平台部署。（1分）

(2) 实现区块链系统中间件服务签名功能启动情况验证。（1分）

(3) 区块链中间件服务节点管理进程启动情况验证和浏览器验证。（1分）

2. 智能合约安全漏洞测试。（7分）

有如下智能合约：

```
pragma solidity ^0.7.6;
contract TimeLock {
    mapping(address => uint) public balances;
    mapping(address => uint) public lockTime;
    function deposit() external payable {
        balances[msg.sender] += msg.value;
        lockTime[msg.sender] = block.timestamp + 1 weeks;
    }
}
```

```

function increaseLockTime(uint _secondsToIncrease) public {
    lockTime[msg.sender] += _secondsToIncrease;
}
function withdraw() public {
    require(balances[msg.sender] > 0, "Insufficient funds");
    require(block.timestamp > lockTime[msg.sender], "Lock time not expired");
    uint amount = balances[msg.sender];
    balances[msg.sender] = 0;
    (bool sent, ) = msg.sender.call{value: amount}("");
    require(sent, "Failed to send Ether");
}
}
contract Attack {
    TimeLock timeLock;
    constructor(TimeLock _timeLock) {
        timeLock = TimeLock(_timeLock);
    }
    fallback() external payable {}
    function attack() public payable {
        timeLock.deposit{value: msg.value}();
        timeLock.increaseLockTime(
            type(uint).max + 1 - timeLock.lockTime(address(this))
        );
        timeLock.withdraw();
    }
}

```

如上代码主要实现功能为规定了转账冻结时间，在冻结时间内用户不能提取存款的金额。

- (1) 分析智能合约中存在问题，并说明危害。（2分）
- (2) 根据测试工具中的代码文件，编写测试用例，复现智能合约中存在的漏洞。（3分）
- (3) 创建新的智能合约，修复其中问题，说明修复内容并测试。（2分）

模块二：智能合约开发与测试（30分）

任务 2-1：智能合约开发（20分）

使用 Solidity 语言完成智能合约开发、部署和调用，要求如下：

1. 供应链金融实体信息编码（6分）

（1）编写供应链金融智能合约的实体接口，完成实体通用数据的初始化，实现企业和票据实体信息上链的功能；（2分）

表 2.2.2.1 SupplyChain 实体说明

名称	类型	说明
companyName	string	公司名称
companyAddress	address	公司地址
creditAsset	uint	信用资产
acceptReceiptIndex	uint[]	接收的凭证
sendReceiptIndex	uint[]	发送的凭证
senderAddress	address	发送票据的地址
accepterAddress	address	接收票据的地址
receiptType	uint8	凭证类型
transferType	uint8	交易类型
amount	uint	交易数量

//公司信息结构体

```
struct Company {  
    //①公司名称  
    //②公司地址  
    //③信用资产  
    //④接收的凭证  
    //⑤发送的凭证  
}
```

//数字发票收据信息

```
struct Receipt {  
    //⑥发送票据的地址  
    //⑦接收票据的地址  
    //⑧凭证类型
```

```
//⑨交易类型
```

```
//⑩交易数量
```

```
}
```

(2) 编写企业上链信息接口，实现供应链金融的企业信息上链；（2分）

```
function addCompany(string name, address companyAddress) returns(bool) {
```

```
//①实例化公司
```

```
//②添加公司地址
```

```
//③将实例化的公司添加到公司映射
```

```
//④返回添加成功标识
```

```
}
```

(3) 基于给定的智能合约代码以及注释，完成银行向企业交易的接口函数；（2分）

```
function bankToCompanyReceipt(address senderAddress, address acceptorAddress,  
uint amount, uint8 receiptType) returns(uint) {
```

①判断接收地址存在

②实例化银行

③实例化公司

```
if (keccak256(bank.bankName) == keccak256("")) {
```

```
    return 404001;
```

```
}
```

```
//确认公司存在
```

```
if (keccak256(company.companyName) == ④) {
```

```
    return 404002;
```

```
}
```

```
if (bank.creditAsset < amount) {
```

```
    return 500001;
```

```
}
```

2. 供应链金融公司与公司接口编码（6分）

(1) 编写公司与公司之间进行交易的历史存证上链接口，实现公司与公司之间的交易功能；（2分）

```

function companyToCompanyReceipt(①, address acceptorAddress, uint amount,
uint8 receiptType) returns(uint) {

    //②接收地址判断
    Company memory senderCompany = companyMap[③];
    Company memory ④ = companyMap[acceptorAddress];
    //确认发送公司存在
    if (keccak256(senderCompany.⑤) == keccak256("")) {
        return 404001;
    }
    //确认接收公司存在
    if (keccak256(acceptorCompany.companyName) == ⑥) {
        return 404002;
    }
    //如果存证接收的公司资产小于存证数额，那么就不能交易发送存证
    if (acceptorCompany.creditAsset ⑦ ⑧) {
        return 500001;
    }
}

```

(2) 编写创建存证的接口，实现创建存证的功能；（2分）

```

Receipt memory newReceipt = Receipt(①, acceptorAddress, receiptType, 2,
amount);

receiptIndex += 1;
//记录存证（存证 Map，公司 Map 对应地址的发送和接收存证列表）
receiptMap[receiptIndex] = ②;
companyMap[③].sendReceiptIndex.push(receiptIndex);
companyMap[acceptorAddress].acceptReceiptIndex.push(④);

```

(3) 编写交易金额数量变化的接口，实现凭证交易双方资金的变化功能；（2分）

```

companyMap[①].creditAsset ② amount;

companyMap[③].creditAsset ④ amount;

```

```
return 200;
```

```
}
```

3. 供应链金融公司与银行交易的接口编码（4分）

（1）编写公司与银行之间进行交易的历史存证上链接口，实现公司与银行之间的交易功能；（2分）

```
function companyToBankReceipt(address senderAddress, ①, uint amount, uint8  
receiptType) returns(uint) {
```

```
    ②
```

```
    Bank memory bank = bankMap[senderAddress];
```

```
    Company memory acceptorCompany = companyMap[③];
```

```
    //确认发送公司存在
```

```
    if (keccak256(bank.bankName) == ④) {
```

```
        return 404001;
```

```
    }
```

```
    //确认接收公司存在
```

```
    if (keccak256(acceptorCompany.companyName) == keccak256("")) {
```

```
        return 404002;
```

```
    }
```

```
    //如果存证接收的公司资产小于存证数额，那么就不能交易发送存证
```

```
    if (acceptorCompany.creditAsset < amount) {
```

```
        return 500001;
```

```
    }
```

（2）编写创建存证的接口，实现创建存证的功能；（1分）

```
//创建存证
```

```
    Receipt memory newReceipt = Receipt(senderAddress, acceptorAddress,  
    ①, 3, amount);
```

```
    receiptIndex ② 1;
```

```
    receiptMap[③] = newReceipt;
```

```
    bankMap[senderAddress].sendReceiptIndex.push(receiptIndex);
```

```
companyMap[accepterAddress].④;
```

(3) 编写交易金额数量变化的接口，实现凭证交易双方资金的变化功能；（1分）

```
bankMap[senderAddress].① ② amount;
```

```
companyMap[accepterAddress].③ ④ amount;
```

```
return 200;
```

```
}
```

4. 合约编译、部署和调用（4分）

(1) 解决代码错误和警告，正确编译并部署合约，成功获取部署的合约地址和 abi。（1分）

(2) 调用食品溯源智能合约的接口，完整验证业务流程。（3分）

任务 2-2：智能合约测试（10 分）

编写智能合约单元测试代码并完成合约功能测试、性能测试，具体要求如下：

1. 配置区块链网络（2分）

启动区块链网络，创建新的 Workspace，配置对外访问的 RPC 接口为 7545，配置项目的配置文件 config.js 实现与新建 Workspace 的连接。

2. 补充给定基础代码中注释提示的部署逻辑（2分）

基于 VSCODE 加载的测试项目，补全位于 test 文件夹中 HelloWorld.js 文件预操作的方法。在测试文件中添加预定义的方法（在其他方法启动前使用）。

3. 补充代码中注释提示的测试逻辑（2分）

基于 VSCODE 加载的测试项目，补全位于 test 文件夹中 HelloWorld.js 文件，添加测试用例，测试智能合约的 get 方法。

4. 测试 hello.get() 方法（2分）

基于 VSCODE 加载的测试项目，补全位于 test 文件夹中 HelloWorld.js 文件，添加测试用例，测试智能合约的 hello.get() 方法。

5. 测试.should.equal 进行对比判断（2分）

基于 VSCODE 加载的测试项目，补全位于 test 文件夹中 HelloWorld.js 文件，添加测试用例，测试智能合约的 equal 字符串比较方法。

模块三：区块链应用系统开发（30 分）

任务 3-1：区块链应用前端功能开发（10 分）

1. 请基于前端系统的开发模板，在注册组件 Register.vue 文件中添加对应的注册逻辑代码，实现对后端系统的注册功能，并测试功能完整性（3 分）：

本题目的具体要求如下：

- (1) 界面有明确的注册相关提示语
- (2) 需要填写的部分有组织名称、区块链地址、组织类型
- (3) 页面需要有“返回”按钮，可以跳转到登录页面
- (4) 点击“注册”按钮时需要检查区块链地址是否已输入
- (5) 注册成功后跳转登录页面

Register.vue:

代码片段 1:

```
<el-row>

  <el-col :span="16" :offset="4">

    <el-form label-width="100px">

      <h3>选手填写部分</h3>

      <el-form-item label="组织名称:">

        <el-input type="primary" v-model="选手填写部分"></el-input>

      </el-form-item>

      <el-form-item label="区块链地址:">

        <el-input type="primary" v-model="选手填写部分"></el-input>

      </el-form-item>

      <el-form-item label="组织类型:">

        <el-radio-group v-model="orgType">

          <el-radio :label="1">公司</el-radio>

          <el-radio :label="2">银行</el-radio>

        </el-radio-group>

      </el-form-item>

    </el-form>

  </el-col>

</el-row>
```

```

        </el-form-item>

    </el-form>

</el-col>

</el-row>

<el-row style="padding-bottom:20px">

    <el-button type="primary" 选手填写部分>注册</el-button>

    <el-button type="primary" 选手填写部分>返回</el-button>

</el-row>

```

代码片段 2:

```

register: function () {

    if (this.address == "") {

        alert(选手填写部分)

    }else {

        let postData = {

            orgType: 选手填写部分,

            username: 选手填写部分,

            address:选手填写部分

        }

        // 和后端交互

        选手填写部分

    }

},

```

代码片段 3:

```

goback: function () {

    this.orgType = ''

    this.username = ''

    this.address = ''

    选手填写部分

}

```

2. 请基于前端系统的开发模板，在登录组件 Login.vue 文件中添加对应的登录逻辑代码，实现对后端系统的登录功能，并测试功能完整性（3 分）：

本题目的具体要求如下：

- (1) 界面有明确的登录相关提示语
- (2) 需要填写的项有用户地址、组织类型
- (3) 页面需要有“注册”按钮，可以跳转注册页面
- (4) 点击“登录”按钮时需要检查各个表项是否已输入
- (5) 登录成功后跳转首页，路径为“/home”

Login.vue:

代码片段 1:

```
<el-col :span="16" :offset="4">

  <el-form label-width="80px">

    <h1>供应链金融应用</h1>

    <h3>选手填写部分</h3>

    <el-form-item label="用户地址:">

      <el-input type="primary" v-model="选手填写部分"></el-input>

    </el-form-item>

    <el-form-item label="组织类型:">

      <el-radio-group v-model="选手填写部分">

        <el-radio :label="1">公司</el-radio>

        <el-radio :label="2">银行</el-radio>

      </el-radio-group>

    </el-form-item>

  </el-form>

</el-col>

</el-row>

<el-row style="margin-bottom: 20px">

  <el-button type="primary" 选手填写部分>登录</el-button>

  <el-button type="primary" 选手填写部分>注册</el-button>

</el-row>
```

```
</el-row>      </el-row>
```

代码片段 2:

```
login: function () {  
    if (this.address == "") {  
        alert("选手填写部分")  
    }else if (this.orgType == "") {  
        alert("选手填写部分")  
    }else {  
        let postData = {  
            orgType: 选手填写部分,  
            address: 选手填写部分  
        }  
        // 与后端交互
```

选手填写部分

```
}
```

```
},
```

代码片段 3:

```
register: function () {  
    选手填写部分  
},
```

3. 请基于前端系统的开发模板，在公司组件 Company.vue 文件中添加对应的逻辑代码，实现对后端系统的公司相关业务功能，并测试功能完整性（2 分）：

Company.vue:

代码片段 1:

```
<el-row>
```

```
<el-col :span="20" :offset="2">
```

```
<el-table :data="companyList" style="font-size: 20px">
```

```
<el-table-column prop="address" label=" 账 户 地 址
```

```
"></el-table-column>
```

```

        <el-table-column prop="name" label="公 司 名 称"
"></el-table-column>

        <el-table-column prop="amount" label="账 户 总 额"
"></el-table-column>

        <el-table-column label="查看详情">

            <template slot-scope="scope">

                <el-button type="primary" @click="选手填写部分">查询
</el-button>

            </template>

        </el-table-column>

        <el-table-column prop="receiptType" label="转账">

            <template slot-scope="scope">

                <el-button type="primary" @click="transfer(scope.row)">操
作</el-button>

            </template>

        </el-table-column>

    </el-table>

</el-col>

</el-row>

<el-row>

```

代码片段 2:

```

<el-dialog title="公司详情" :visible.sync="dialogVisible">

    <el-form label-width="100px">

        <el-form-item label="账户地址:">

            {{选手填写部分 }}

        </el-form-item>

        <el-form-item label="公司名称:">

            {{ 选手填写部分 }}

        </el-form-item>

```

```

    <el-form-item label="账户总额:">

      {{ 选手填写部分 }}

    </el-form-item>

  </el-form>

```

代码片段 3:

```

detail: function (queryAddress) {

  this.dialogVisible = true

  let address = 选手填写部分

  this.axios.get(`选手填写部
分?address=${address}&queryAddress=${queryAddress}`)

    .then((response) => {

      console.log(response)

      if (response.data.code == 200) {

        let inAddress = response.data.data.companyV0.address;

        let inName = 选手填写部分;

        let inAmount = response.data.data.companyV0.amount;

        this.companyDetail = {

          address: 选手填写部分,

          name: 选手填写部分,

          amount: 选手填写部分,

          senderReceiptList: response.data.data.senderReceiptList,

          acceptorReceiptList: response.data.data.acceptorReceiptList

        }

      } else {

        alert(`请求内容有误, ${response.data.data}`)

      }

    })

},

```

4. 请基于前端系统的开发模板，在银行组件 Bank.vue 文件中添加对应的逻辑代码，实

现对后端系统的银行相关业务功能，并测试功能完整性（2分）：

Bank.vue:

代码片段 1:

```
<el-row>

  <el-dialog title="交易(发送凭证)页":visible.sync="transDialogVisible"
width="30%">

    <el-form label-width="100px">

      <el-form-item label="发送账户地址:">

        {{选手填写部分}}

      </el-form-item>

      <el-form-item label="接收账户地址:">

        {{选手填写部分}}

      </el-form-item>

      <el-form-item label="交易额:">

        <el-col :span="16" :offset="4">

          <el-input type="primary" v-model="选手填写部分"></el-input>

        </el-col>

      </el-form-item>

      <el-form-item label="凭证类型:">

        <el-select v-model="选手填写部分" placeholder="请选择">

          <el-option

            v-for="item in options"

            :key="item.value"

            :label="item.label"

            :value="item.value">

          </el-option>

        </el-select>

      </el-form-item>

    </el-form>
```

```

        <el-row>

            <el-button type="primary" size="medium" @click="选手填写部分">确定
        </el-button>

    </el-row>

</el-dialog>

</el-row>

```

代码片段 2:

```

executeTransaction: function() {

    let funcName = "companyToBankReceipt";

    if (this.transDetail.amount ==选手填写部分) {

        alert('交易额不能为空! ')

        return

    }

    if (this.$cookies.get('orgType') == 选手填写部分) {

        alert('银行不能给银行发送凭证! ')

        return

    }

    if (选手填写部分) {

        alert("凭证发送账户和接收账户不能相一致! ")

        return

    }

    this.axios.post(`/finance/transaction/${funcName}`, 选手填写部
分).then((response) => {

        if (response.data.code == 200) {

            alert('凭证发送成功')

            this.query()

            this.transDialogVisible = false

        } else {

            alert(`凭证发送失败, ${response.data.data}`)

```

```
    }  
    })  
    },
```

任务 3-2：区块链应用后端功能开发（20 分）

1. 开发区块链供应链金融应用后端系统中用户功能模块对应的用户注册功能,根据前后代码补充最合适的代码,并测试功能完整性。（4 分）

OrgServiceImpl.java:

```
/**  
 * 注册 Service  
 * RegisterBO registerBO  
 * */  
@Override  
public Result<String> register(RegisterBO registerBO) {  
    if (StrUtil.isEmpty(选手填写部分) ||  
        StrUtil.isEmpty(选手填写部分) ||  
        registerBO.getOrgType() == 选手填写部分  
    ) {  
        return Result.error(ResultVO.PARAM_EMPTY);  
    }  
    List funcParam = new ArrayList();  
    funcParam.add(选手填写部分);  
    funcParam.add(选手填写部分);  
    if(registerBO.getOrgType() == 2){  
        funcParam.add(BigInteger.valueOf(1000));  
    }  
    String funcName;  
    if(registerBO.getOrgType() == 2){  
        funcName =选手填写部分;
```

```

        }else{

            funcName =选手填写部分;

        }

        String _result =
weBASEUtils.funcPost(OWNER_ADDRESS,funcName,funcParam);

        JSONObject _resultJson = JSONUtil.parseObj(_result);

        if      (_resultJson.containsKey("statusOK")      ==      false      ||
_resultJson.getBool("statusOK") != true){ // _resultJson.getInt("code") > 0

            return Result.error(ResultVO.选手填写部分);

        }

        return Result.success("ok");

    }

```

2. 开发区块链供应链金融应用中后端系统中用户功能模块对应的用户登录功能,根据前后代码补充最合适的代码,并测试功能完整性。(4分)

OrgServiceImpl.java:

```

/**
 * 登录 Service
 * LoginBO loginBO
 **/

@Override

public Result<String> login(@RequestBody LoginBO loginBO) {

    if (StrUtil.isEmpty(loginBO.getAddress())

    ) {

        return Result.error(ResultVO.PARAM_EMPTY);

    }

    List funcParam = new ArrayList();

    funcParam.add(选手填写部分);

    String funcName;

    if(loginBO.getOrgType() == 2){

```

```

        funcName =选手填写部分;

    }else{

        funcName =选手填写部分;

    }

    String _result = weBASEUtils.funcPost(选手填写部
分,funcName,funcParam);

    JSONArray _resultJson = JSONUtil.parseArray(_result);

    if (StrUtil.isEmpty(_resultJson.get(0).toString())){

        return Result.error(ResultV0.选手填写部分);

    }

    return Result.success("ok");

}

```

3. 开发区块链供应链金融应用后端系统中查询功能模块对应的查询所有公司信息功能，根据前后代码补充最合适的代码，并测试功能完整性。（4分）

```

QueryServiceImpl.java:

/**
 * 获取所有公司数据，不包含存证详细信息
 * */
@Override
public Result listCompany(String userAddress){

    String _result =
weBASEUtils.funcPost(userAddress,"getAllCompanyAddress",new ArrayList());

    try {

        JSONArray respArray = 选手填写部分;

        String a = respArray.get(0).toString();

        JSONArray addressArray = 选手填写部分;

        List<CompanyV0> companyList = new ArrayList<>();

        for(Object obj: addressArray) {

            CompanyV0 companyV0 = 选手填写部分;

```

```

        companyList.add(companyVO);
    }

    return Result.success(选手填写部分);
} catch (Exception e) {

    ResultVO resultVO = ResultVO.CONTRACT_ERROR;

    resultVO.setData(选手填写部分);

    return Result.error(选手填写部分);

}
}

```

4. 开发区块链供应链金融应用的后端系统中查询功能模块对应的查询银行凭证列表功能根据前后代码补充最合适的代码，并测试功能完整性。（4分）

QueryServiceImpl.java:

```

/**
 * 获取包括银行详情，银行发送凭证列表，银行接收凭证列表
 * */
@Override
public Result getBankEntity(String userAddress, String queryAddress) {

    BankVO bankVO = 选手填写部分;

    List<ReceiptVO> senderReceiptList = 选手填写部分;

    List<ReceiptVO> acceptorReceiptList = 选手填写部分;

    BankEntityVO bankEntityVO = new BankEntityVO();

    bankEntityVO.setBankVO(选手填写部分);

    bankEntityVO.setSenderReceiptList(选手填写部分);

    bankEntityVO.setAcceptorReceiptList(选手填写部分);

    return Result.success(bankEntityVO);

}

```

5. 开发区块链供应链金融应用的后端系统中交易功能模块对应的银行向公司交易凭证功能，根据前后代码补充最合适的代码，并测试功能完整性。（2分）

TransactionServiceImpl.java:

```

/**
 * 公司向银行发送凭证（银行转账给公司）
 * */
public Result<String> bankToCompanyReceipt(TransactionBO transactionBO) {
    选手填写部分;
}

```

6. 开发区块链供应链金融应用的后端系统中交易功能模块对应的公司向公司交易凭证功能，根据前后代码补充最合适的代码，并测试功能完整性。（2分）

TransactionServiceImpl.java:

```

/**
 * 公司向公司的凭证发送
 * */
public Result<String> companyToCompanyReceipt(TransactionBO
transactionBO) {
    选手填写部分;
}

```

（四）任务书生成

本赛项采用公开样题的赛卷形式，专家组编制完成样题后，由大赛执委会在省赛平台正式发布。赛项样题将于正式比赛前 30 天以上发布。正式比赛试题由专家组依据竞赛规程和样题模式进行编制。

正式比赛试题由专家组依据竞赛规程和样题模式进行编制，共编制 1 套正式比赛试题，试题的重复率不超过 30%。

五、竞赛方式

（一）竞赛形式

线下比赛。

（二）组队方式

竞赛以团体赛方式进行，以院校为单位参赛，每支参赛队由 2 名选手组成，不得跨校组队，同一学校参赛队不超过 3 队。参赛队可配指导教师，指导教师须为本校专兼职教师，每队限报 2 名指导教师，竞赛期间不允许指导教师进入赛场进行现场指导。指导教师负责参赛选手的报名、训练指导、服务和比赛期间参赛选手的日常管理。

（三）参赛选手资格

参赛选手须为高等职业学校专科、高等职业学校本科全日制在籍学生（以报名时的学籍信息为准）、五年制高职（四、五年级）在籍注册学生。参赛选手年龄须不超过 25 周岁，年龄计算的截止时间以竞赛当年规定为准。凡在往届全省职业院校技能大赛中获一等奖的选手，不能再参加同一项目同一组别的比赛。

（四）人员变更

参赛选手获得确认后不得随意更换。如参赛选手因故无法参赛，须出具书面说明，经大赛执委会办公室核实后予以更换，补充人员需满足本赛项参赛选手资格并接受审核。

（五）熟悉场地

参赛队可于正式比赛 1 天前，由主办方统一组织熟悉场地。比赛赛位通过抽签决定，参赛选手在赛前 10 分钟领取比赛任务，并进入比赛赛位，比赛正式开始后方可进行相关

操作。比赛期间参赛选手原则上不得离开比赛场地。选手分工、工作程序和时间安排由参赛队自行决定。

(六)入场规则

参赛选手凭参赛证、身份证、学生证在正式比赛开始前规定时间内到达指定地点集合，选手依次进场，进行各项准备工作，现场裁判将对各参赛选手的身份信息进行核对。

(七)赛场规则

参赛选手进入赛场必须听从裁判员统一指挥，需对比赛设备进行 检查，如有问题及时向裁判报告。参赛选手必须在裁判宣布比赛开始后才能进行比赛。比赛过程中选手不得随意离开工位，不得与其他选手交流或擅自离开赛场，如遇问题须举手向裁判示意询问后处理，否则视为作弊。比赛过程中只允许裁判员、工作人员进入现场，其余人员未经裁判长 同意不得进入赛场。比赛过程中，选手严格遵守安全操作规程，确保人身和设备安全，因选手原因造成设备故障或损坏，无法继续比赛，裁判长有权决定终止其比赛。因非选手因素造成设备故障，由裁判长视具体情况做出裁决。

(八)离场规则

竞赛结束，选手必须停止一切操作。参赛队若提前结束竞赛，选手向裁判举手示意，裁判记录终止时间，结束后选手不得再进行任何操作。

(九)成绩评定与结果公布

裁判依据赛项评分标准对参赛选手竞赛过程及成果进

行评定。赛项的最终总成绩经裁判长、监督员审核无误后签字，并将签字的纸质成绩单进行公示，并报送大赛执委会办公室。

六、竞赛环境

1. 竞赛场地：竞赛场地分为竞赛现场、裁判员休息区、指导老师休息区，竞赛现场设置竞赛区、现场裁判工作区和技术支持区，以上区域应保证采光、照明和通风良好。

2. 竞赛设备：场内竞赛区按照参赛队数量准备比赛所需的软硬件平台，为参赛队提供统一竞赛设备和备用设备，选手无需自带任何工具及材料。

3. 竞赛工位：竞赛现场每个参赛队工作区间面积合理，确保参赛队之间互不干扰，每个参赛队工作区上标明编号，各参赛队工作区配备独立电源。

4. 技术支持区：为技术支持人员提供固定工位、电源保障。技术支持保障人员在技术支持服务区候场，有需要时在现场裁判的带领下到相关的工位进行赛场技术支持保障。

5. 竞赛现场符合消防安全规定，现场消防器材和消防栓合格有效，应急照明设施状态合格，赛场明显张贴紧急疏散图，赛场地面张贴疏散指示箭头，赛场出入口专人负责，随时保证安全通道的畅通无阻。

七、技术规范

按照《全国职业院校技能大赛赛项规程编制要求》，结合企业职业岗位对人才培养需求，并参照相关国家职业标准

制定。参赛代表队在实施竞赛项目中要求遵循的规范见表 4。

表 4 技术规范

序号	标准号	内容
1	GB/T 11457-2006	信息技术、软件工程术语
2	LD/T81.1-2006	职业技能实训和鉴定设备技术规范
3	GB/T 25069-2010	信息安全技术 术语
4	ISO 22739-2020	Blockchain and distributed ledger technologies — Vocabulary（区块链和分布式账本技术词汇）
5	GBZ2-02-10-15	区块链工程技术人员国家职业技能标准
6	GBZ4-04-05-06	区块链应用操作员国家职业技能标准
7	CBD-Forum-001-2017	区块链参考架构
8	CBD-Forum-002-2017	区块链数据格式规范
9	T/SIA 007-2018	区块链平台基础技术要求
10	CIET-2018-04	区块链技术人才培养标准

八、技术平台

表 5 硬件设备

设备名称	数量	备注
服务器	1	处理器 i7/64G 内存/固态硬盘 1T 及以上、USB3.0 接口、千兆及以上网卡
路由交换模块	1	通用网络设备
PC	2 台/队	通用台式机 处理器 i5/16G 内存/固态硬盘 512G 及以上、USB3.0 接口、千兆及以上网卡

表 6 服务端软件

序号	软件	备注
----	----	----

1	区块链服务端操作系统	Ubuntu 20.04.X
2	区块链底层平台	FISCO BCOS 2.8.0
3	区块链竞赛平台	区块链竞赛通用平台

表 7 PC 工具软件

序号	软件类别	软件名称、版本号
1	操作系统	Windows 10 (64 位)
2	浏览器	Chrome
3	开发语言	JAVA 8
4		Solidity 0.4.25 及以上
5		JavaScript
6	开发工具	IDEA 2022 (Community Edition) 及以上
7		Visual Studio Code 1.69 及以上
8	构建工具	Maven 3.8.1 及以上
9	SSH 工具	Putty
10	接口测试工具	Postman
11	前端框架	Vue.js 2.*
12	绘图工具	Office Visio 2013 及以上
13	截图工具	Windows 系统自带
14	文档编辑器	WPS 2019 及以上
15	输入法	搜狗拼音输入法
16	虚拟化工具	VMware

九、评分办法

(一) 评分原则

1. 竞赛评分严格遵守公开、公平、公正、独立、透明的原则，区块链技术应用赛项评分采用赛项结果评分方法，赛项最终得分按百分制计算，贯彻落实大赛坚持的公平、公正和公开原则。

2. 赛项合作企业不得直接或者间接地参与赛项评分。

3. 赛项评分依据选手固化在实操任务中的成果，通过评分裁判对比赛成果再现的方法评分，并兼顾团队协作精神和职业素养综合评定。

4. 为了确保赛事评判的客观性，制定详细的评分标准，细化评分项目，尽可能量化每一评分项目的评分标准，减少主观判断比例，确保赛事客观公正。

5. 评分过程全程可追溯。

（二）评分标准

表 8 评分标准

赛项模块	竞赛内容	考核的知识点、技能点	相应得分点	分值
模块一 区块链网络部署	区块链系统部署与运维	基于给定的环境和区块链系统，完成区块链系统部署及节点部署。通过监控工具完成对网络、节点服务的监控。根据业务需求规范，完成系统日志、网络参数、节点服务等系统结构的维护	1. 按要求正确部署区块链网络，并能验证运行状态 2. 正确安装管理工具及监控工具 3. 正确完成业务系统数据库的创建和管理 4. 搭建的区块链符合业务需求，按要求进行扩容和网络配置等维护操作	25
	区块链系统测试	设计区块链系统的测试流程，调用智能合约进行单元测试、集成测试、系统测试和性能测试；根据业务需求，分析并修复给定智能合约中的安全漏洞	1. 对测试需求分析正确，合理设计测试用例 2. 正确对区块链系统进行防篡改测试、签名测试等 3. 正确使用测试工具修复合约中的漏洞 4. 正确对已部署的智能合约进行性能测试、系统测试和执行分析	10
模块二 智能合约开发与测试	智能合约开发	使用 Solidity 语言进行智能合约开发，完成智能合约部署和调用	1. 合约编写功能覆盖全面、逻辑正确 2. 正确部署和调用合约	20
	智能合约	编写智能合约单元测试	1. 对已有合约正确构建	10

赛项模块	竞赛内容	考核的知识点、技能点	相应得分点	分值
试	测试	代码并完成合约功能测试、性能测试	单元测试 2. 正常使用工具完成合约功能及性能测试	
模块三 区块链应用系统开发	区块链应用前端开发	根据业务需求,使用前端开发框架完成页面设计,使用已提供的服务端接口获取业务数据,并进行部署展示	1. 正确编写前端代码,完成服务端接口调用 2. 正确完成前端数据展示及页面逻辑	10
	区块链应用后端开发	依据功能需求,使用 Java 等后端开发语言及常用框架进行后端代码开发,访问数据库、实现应用程序接口、调用智能合约,完善区块链应用系统,完成后端代码的部署	1. 正确使用后端开发语言和框架,完成数据库调用等功能,实现应用程序接口 2. 正确编写后端接口程序,调用智能合约,实现链上信息的查询和结果展示 3. 正确编写后端接口程序,进行区块链应用操作 4. 正确部署后端程序	20
职业素养		文档写作科学规范,具有团队合作精神和创新意识,比赛操作严谨,代码编写规范,文明竞赛	1. 分工合理 2. 操作规范 3. 文明竞赛	5

(三) 评分方式

1. 参与赛项成绩管理的组织机构包括裁判组、仲裁组和监督组,裁判组实行“裁判长负责制”,设裁判长1名,全面负责赛项的裁判与管理工作,裁判组包含评分裁判、加密裁判和现场裁判。

2. 评分裁判负责对参赛队的竞赛成果按赛项评分标准进行评定。在比赛进行的过程中评分裁判不到比赛现场,参赛选手退出赛场后,评分裁判进入现场对各工位进行评分。

3. 赛项采取两次加密原则,加密裁判负责对参赛选手抽

签进行加密并且保密，不得将任何信息透露给其他人员，否则按照相关规定予以处理。

4. 现场裁判负责对整个赛场进行巡查和监督，必须严格按照现场裁判要求做好相应工作。

5. 监督组对裁判组的工作进行全程监督，并对竞赛成绩抽检复核。仲裁组负责接受由参赛队领队提出的书面申诉，组织复议并及时反馈复议结果。

6. 赛项成绩解密后，经裁判长、监督组签字后，在大赛办指定的地点，以纸质形式向全体参赛队进行公布。成绩公布 2 小时无异议后，将赛项总成绩的最终结果录入赛务管理系统。

十、奖项设定

根据参赛队竞赛成绩排名分别设立一、二、三等奖。以各赛项实际参赛队数量为基数，一、二、三等奖获奖比例分别为 10%、20%、30%（小数点后四舍五入）。其他情况按照竞赛规程总则执行。

十一、申诉与仲裁

1. 各参赛队对不符合赛项规程规定的仪器、设备、工装、材料、物件、计算机软硬件、竞赛使用工具、用品，竞赛执裁、赛场管理、竞赛成绩，以及工作人员的不规范行为等，可向赛项仲裁组提出申诉，申诉主体为参赛队领队。

2. 申诉启动时，参赛队向赛项仲裁组递交领队亲笔签字的书面报告。书面报告应对申诉事件的现象、发生时间、涉

及人员、申诉依据等进行充分、实事求是的叙述。非书面申诉不予受理。

3. 提出申诉的时间应在比赛结束后(选手赛场比赛内容全部完成)2 小时内。超过时效不予受理。

4. 赛项仲裁组在接到申诉报告后的 2 小时内组织复议,并及时将复议结果以书面形式告知申诉方。申诉方对复议结果仍有异议,可由领队向比赛监督员提出申诉,由监督员传达最终仲裁结果。

5. 申诉方不得以任何理由拒绝接收仲裁结果,不得以任何理由采取过激行为扰乱赛场秩序。仲裁结果由申诉人签收,不能代收,如在约定时间和地点申诉人离开,视为自行放弃申诉

6. 申诉方可随时提出放弃申诉。

十二、安全预案

(一) 竞赛环境突发应急预案

1. 赛场备用工位: 赛场提供占总参赛队伍 10%的备用工位。

2. 竞赛系统可靠性: 竞赛系统使用的服务器应进行冗余,数据库、存储应使用高可用架构。

3. 服务器资源问题: 若服务器在比赛过程中出现卡顿、死机等情况,在现场裁判与技术人员确定情况后,可更换服务器资源。

4. PC 机问题: 若 PC 机在比赛过程中出现死机、蓝屏等

现象（重启后无法解决），在现场裁判与技术人员确定情况后，可启用备用工位或更换 PC 机。

（二）参赛选手突发情况应急预案

竞赛期间，安排 2 名医生现场值班，安排好车辆随时待命，一旦参赛选手出现发病、受伤、意外伤害或者食品安全事故，需及时对选手进行现场救治或送医治疗。

（三）停水停电及火情突发应急预案

1. 承办校后勤部门须保证竞赛期间供电、供水正常遇停电及时启用自备电源（或租赁发电机）供电，保证竞赛设备用电正常。

2. 发现火情时立即根据火情状况，决定是否组织人员疏散，是否切断电源和光源，以及是否需要报警。

（四）治安事件突发应急预案

规范赛场秩序，加强法制和安全教育，对发现有情绪异常、行为过激的选手，及时与参赛队领队沟通联系，做好劝导和化解工作。

十三、其他规定

（一）参赛队须知

1. 参赛队名称需统一使用规定的学校代表队名称，不使用其他组织、团体的名称。

2. 各参赛院校应指定 1 名负责人任赛项领队，全权负责参赛事务的组织、协调和领导工作。

3. 参赛队按照赛项赛程安排，凭参赛证、身份证或护照、

学生证参加比赛及相关活动。

4. 主办方统一安排各参赛队在比赛前一天进入赛场熟悉环境和设施情况。

5. 参赛队选手、领队和指导教师要有良好的职业道德，严格遵守比赛规则和比赛纪律，服从裁判，尊重裁判和赛场工作人员，自觉维护赛场秩序。

6. 领队应负责赛事活动期间本队所有选手的人身及财产安全，如发现意外事故，应及时向大赛办报告。

7. 各学校组织代表队时，应为参赛选手购买大赛期间的人身意外伤害保险。

（二）领队、指导教师须知

1. 严格遵守赛场的各项规定，服从裁判，文明竞赛。如发现弄虚作假者，取消参赛资格，名次无效。

2. 领队和指导教师务必带好有效身份证件，在活动过程中佩戴“领队证”“指导教师证”参加竞赛相关活动。

3. 领队要严格遵守竞赛的各项规定，加强对参赛人员的管理，做好赛前准备工作，督促选手带好证件等竞赛相关材料。

4. 在比赛期间要严格遵守比赛规则，不得私自接触裁判人员。

5. 竞赛过程中，未经裁判许可，领队、指导教师及其他人员一律不得进入竞赛现场。

6. 如对竞赛过程有疑议，由领队负责以书面形式向大赛

仲裁组反映，但不得影响竞赛进行。

7. 对申诉的仲裁结果，领队要带头服从和执行，并做好选手工作。参赛选手不得因申诉或对处理意见不服而停止竞赛，否则以弃权处理。

8. 领队和指导老师应及时查看有关赛项的通知和内容，认真研究和掌握本赛项竞赛的规程、技术规范和赛场要求，指导选手做好赛前的一切技术准备和竞赛准备。

（三）参赛选手须知

1. 参赛选手应严格遵守赛场规章、操作规程和工艺准则，保证人身及设备安全，接受裁判员的监督和警示，文明竞赛。

2. 选手应按照规定时间抵达赛场，凭统一发放的参赛证、身份证或护照、学生证完成入场检录、抽签确定竞赛工位号，不得迟到早退。

3. 参赛选手进入赛场前，须将身份证、学生证交由检录人员统一保管，不得带入场内。参赛证始终佩戴，以备检查。

4. 参赛选手凭竞赛工位号进入赛场，不允许携带任何书籍和其他纸质资料，竞赛统一提供草稿纸。不允许携带任何电子设备及通信工具和存储设备（如U盘），竞赛统一提供计算机以及应用软件。

5. 参赛选手应在规定的时间段进入赛场，认真核对竞赛工位号，在指定位置就座。入场后，赛场工作人员与参赛选手共同确认操作条件及设备状况，填写相关确认文件，并由参赛队长确认签字（签竞赛工位号）。

6. 参赛选手在收到开赛信息前不得启动操作。在竞赛过程中，确因计算机软件或硬件故障，致使操作无法继续的，经裁判长确认，予以启用备用计算机。

7. 参赛选手应在竞赛规定的时间完成任务书内容，并按要求提交成果物。

8. 参赛选手需及时保存工作记录。对于因自身原因造成的数据丢失，由参赛选手自行负责。

9. 参赛队所提交的答卷采用竞赛工位号进行标识，不得出现地名、校名、姓名、参赛证编号等信息，否则取消竞赛成绩。

10. 竞赛过程中，因严重操作失误或安全事故不能进行比赛的，现场裁判员有权中止该队比赛。

11. 参赛期间，食品、饮水等由赛场统一提供，选手休息和如厕时间计算在比赛时间内。

12. 在参赛期间，选手应注意保持工作环境及设备摆放符合企业生产“5S”（即整理、整顿、清扫、清洁和素养）要求。

13. 在比赛中如遇非人为因素造成的设备故障，经裁判确认后，可向裁判长申请补足排除故障的时间。

14. 参赛选手原则上不得提前结束比赛。如确因不可抗力因素需要提前结束比赛的，须向现场裁判员举手示意，经裁判长许可并完成记录后，方可离开。

15. 竞赛时间结束，全体选手应立刻起立，结束操作，

将资料 and 工具整齐摆放在操作平台上，将竞赛成果物按照赛题指定的方式提交，并由现场裁判和参赛队长共同签字确认后，方可离开赛场，离开赛场时不得带走任何资料。

16. 参赛选手未能按时提交竞赛成果物的，竞赛成绩计为零分。

17. 在竞赛期间，未经赛项执委会批准，参赛选手不得接受其他单位和个人进行的与竞赛内容相关的采访。参赛选手不得将竞赛的相关信息私自公布。

18. 符合下列情形之一的参赛选手，经裁判组裁定后终止其竞赛：

（1）不服从裁判员管理、扰乱赛场秩序、干扰其他参赛选手比赛，裁判员应对其提出警告。二次警告后无效，或情节特别严重造成竞赛中止的，经裁判长确认，终止其比赛，并取消竞赛资格和竞赛成绩。

（2）竞赛过程中，如选手人为造成计算机、仪器设备及工具等严重损坏，负责赔偿其损失，裁判长有权裁定其结束竞赛、取消竞赛成绩或取消竞赛资格。

（3）竞赛过程中，造成重大安全事故，或产生重大安全隐患，经裁判员提示没有采取措施的，裁判员可暂停其竞赛，由裁判长裁定其结束竞赛、取消竞赛成绩或取消竞赛资格。

（四）工作人员须知

1. 树立服务观念，注意文明礼貌，全程佩带证件，保持

良好形象。

2. 认真学习赛项指南，以高度负责的精神、严肃认真的态度和严谨细致的作风，认真履行岗位职责。

3. 在大赛办的领导下，服从调配和分工，确保比赛工作的顺利进行。

4. 自觉遵守赛项纪律和规则，保守秘密。

5. 严守工作岗位，按时到岗，不无故离岗，特殊情况需向大赛办请假。

6. 熟悉应急预案，如遇突发事件，及时组织疏散，确保人员安全。

7. 工作人员在比赛中若有舞弊行为，立即撤销其工作资格，并按规定严肃处理。

8. 保持沟通，加强协作，提高工作效率。

（五）竞赛观摩

本赛项可根据具体情况安排相关院校、行业组织，并组织相关校企进行交流研讨活动。