2019年全国职业院校技能大赛

高职组“区块链技术应用”

赛项任务书（样书）

2019年8月

第一部分 竞赛须知

一、竞赛要求

1. 正确使用工具，操作安全规范；
2. 竞赛过程中如有异议，可向现场裁判反映，不得扰乱赛场秩序；
3. 遵守赛场纪律，尊重裁判，服从安排。

二、职业素养与安全意识

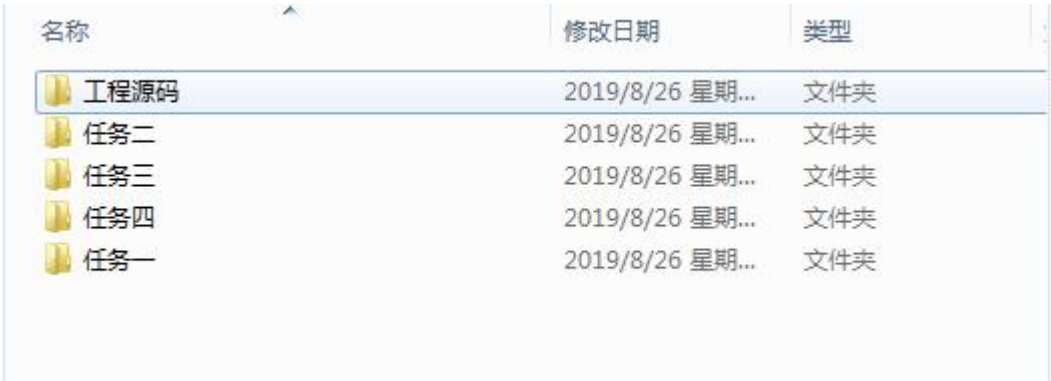
1. 完成竞赛任务所有操作符合安全操作规范，注意用电安全；
2. 操作台、工作台表面整洁；
3. 遵守赛场纪律，尊重赛场工作人员；爱惜赛场设备、器材。

三、扣分项

1. 在竞赛过程中，因操作不当导致设备破坏性损坏或造成事故，视情节扣 10～20 分， 情况严重者取消比赛资格；
2. 衣着不整、污染赛场环境、扰乱赛场秩序、干扰裁判工作等不符合职业规范的行为， 视情节扣 5～10 分，情节严重者取消竞赛资格；
3. 竞赛过程中，解答题目如果出现使用虚假数值、随机数值仿冒真实采集到的数值充当竞赛结果误导裁判，一经核实代码后，本道题不得分，总分倒扣 3 至 5 分。

四、选手须知

1. 任务书如出现缺页、字迹不清等问题，请及时向裁判示意，并进行更换；比赛结束后，比赛提供的所有纸质材料等不得带离赛场；
2. 竞赛内容编写请严格按照任务书的要求进行规范操作；
3. 参赛团队应在规定时间内完成任务书要求的内容，任务实现过程中形成的文件资料必须存储到的指定文件夹位置上传任务平台，竞赛资料存储在“区块链竞赛”文件夹中。未存储到指定位置的文件夹均不得分；
4. 选手创建文件夹，任务一存放在“区块链竞赛/任务一”文件夹路径下，任务二存放在“区块链竞赛/任务二”文件夹路径下，任务三存放在“区块链竞赛/任务三”文件夹路径下，任务四存放在“区块链竞赛/任务四”文件夹下，把任务一和任务二添加至工程源码中，保存在“区块链竞赛/工程源码”，所有任务完成后，上传区块链竞赛文件夹至任务平台。
5. 比赛过程中由于人为操作失误造成器件损坏，器件不予更换；
6. 在裁判长宣布竞赛结束后，参赛选手应立即停止对竞赛设备与计算机的任何操作。
7. 任务书中只得填写竞赛相关信息，不得出现学校、姓名等与身份有关的信息或与竞赛过程无关的内容，否则成绩无效。
8. 竞赛过程中严禁更改竞赛平台各单元内部电路、气路接线。由于参赛选手人为原因导致竞赛设备损坏，以致无法正常继续比赛，将取消参赛队竞赛资格。



竞赛场次：第 场 赛位号：第 号

第二部分 竞赛设备及注意事项

赛场提供区块链技术应用设备一套, 竞赛选手依照本竞赛项目的任务内容，完成任务书要求的相关操作与开发任务。

注意事项

1. 检查硬件设备、电脑设备是否正常。检查竞赛所需的各项设备、软件和竞赛材料等；
2. 竞赛任务中所使用的各类软件工具、软件安装文件等，安装好，请自行根据竞赛任务要求使用；
3. 竞赛过程中请严格按照竞赛任务中的描述，对各区块链设备进行操作使用，对于与竞赛无关的任务代码，请勿变动，防止系统报错。
4. 竞赛任务完成后，需要保存，不要关闭任何设备，不要拆动硬件的连接线，不要对设备随意加密。

第三部分 竞赛任务

一、任务描述

在传统的生鲜食品供应系统中，对于运送的温度变化，无法做到良好的监控，导致出现了货品的损坏，到最后三方陷入到关于赔偿的纠纷讨论，影响了生态的良好发展。因此运送途中的温度真实问题，供应商和物流签订好相应合同，就变得极具意义。所以现在do东超市委托我公司开发区块链生鲜食品系统，要求能够有查看物流，查看实时温度功能，请开发人员实现以下功能。

二、任务要求

任务一：编写链码

一、比赛要求

请参赛选手在规定时间内，实现Init函数，实现Invoke函数和完成试题中指定的四项功能函数。

完成后将任务文件打包存放在“区块链竞赛/任务一”文件夹路径下。

二、比赛内容

1. 实现Init函数

```
func (t *PerishableFood) Init(stub shim.ChaincodeStubInterface) pb.Response {

}。
```

2. 实现Invoke函数 由于实现的功能较多，会根据传入的参数来区分实际应该调用的具体链码。

```
func (t *PerishableFood) Invoke(stub shim.ChaincodeStubInterface) pb.Response {

}
```

3. 实现创建商品 创建商品的过程分为以下几部分。 创建商品主键、检查是否存在、写入商品数据 具体实现：

    1. 检查参数个数（商品所需的所有数据）
    2. 验证参数是否为空
    3. 通过商品ID创建主键
    4. 验证商品是否存在
    5. 转换温度、价格的数据类型
    6. 构建商品数据
    7. 序列化对象（转成json数组）
    8. 写入区块链

```
func createCommodity(stub shim.ChaincodeStubInterface, args []string) pb.Response {

}
```

4. 创建订单 创建订单的过程分为以下几部分。 获取商品信息、构建订单对象、创建订单主键、写入订单数据 具体实现：

    1. 检查参数个数（订单所需的所有数据）
    2. 验证参数是否为空
    3. 通过商品ID创建主键
    4. 通过主键获取商品数据
    5. 将数据包装成商品对象
    6. 转换时间、实例的数据类型
    7. 构建订单数据
    8. 序列化对象
    9. 通过订单ID创建主键
    10. 写入区块链

```
func createOrder(stub shim.ChaincodeStubInterface, args []string) pb.Response {

}
```

5. 查询商品列表 查询商品列表的过程分为以下几部分。 获取所有使用commodity作为前缀的主键的数据，并构建成json对象返回 具体实现：
    1. 检查参数个数（查询不需要任何数据）
    2. 通过主键从区块链查找相关的数据
    3. 遍历所有商品，添加进列表
    4. 序列化数据并返回

```
func queryCommodityList(stub shim.ChaincodeStubInterface, args []string) pb.Response {

}
```

6. 查询订单列表 查询订单列表的过程分为以下几部分。 获取所有使用order作为前缀的主键的数据，并构建成json对象返回 具体实现：
    1. 检查参数个数
    2. 通过主键从区块链查找相关的数据
    3. 遍历订单数据，添加进列表
    4. 序列化数据并返回

```
func queryOrderList(stub shim.ChaincodeStubInterface, args []string) pb.Response {

}
```

任务2：编写程序后台

一、比赛要求

参赛选手在规定的时间内，实现路由初始化setupRouter()函数，实现main()函数和编写应用程序后端，实现题目指定接口。

完成后将任务文件打包存放在“区块链竞赛/任务二”文件夹路径下

二、比赛内容

1. 实现路由初始化setupRouter()函数 编写一个设置路由的函数setupRouter。 router.，其中method即是该路径允许的方法，第一个参数是路径，第二个参数是handle这个请求的中间件。 地址：perishable-food\application\main.go

```
// 设置路由
func setupRouter() *gin.Engine{

}
```

2. 实现main()函数 在main函数中设置路由然后启动即可，路由中包含了我们需要监听的地址，当地址符合时，调用对应的控制方法。 地址：perishable-food\application\main.go

```
func main(){

}
```

3. 创建商品 思路：
    1. 将前端提交的商品的参数传入链码调用createCommodity函数
    2. 创建商品的请求体
    3. 解析前端提交的数据并实例化对象

4. 调用channelExecute执行createCommodity，将对象的参数转成byte数组传入
5. 将链码执行的结果以json格式应答前端请求

```
// 创建商品
func createCommodity(ctx *gin.Context){3

}
```

4. 查询商品列表 思路：

1. 调用链码的queryCommodityList函数
2. 调用channelQuery执行queryCommodityList，传入一个空数组
3. 将链码执行的结果反序列化应答给前端

```
// 查询商品列表
func commodityList(ctx *gin.Context){3

}
```

5. 创建订单 思路：

1. 将前端提交的订单的参数传入链码调用createOrder函数
2. 创建订单的请求体
3. 解析前端提交的数据并实例化对象
4. 调用channelExecute执行createOrder，将对象的参数转成byte数组传入
5. 将链码执行的结果以json格式应答前端请求

```
// 创建订单
func createOrder(ctx *gin.Context){3

}
```

6. 查询订单列表 思路：

1. 将前端提交的订单ID传入链码调用queryOrderList函数
2. 创建订单的应答体（参照链码中的订单定义）
3. 检查请求中有无orderId数据，如果有则查询单个订单，没有则查全部订单
4. 调用channelQuery执行queryOrderList，传入查询订单的ID
5. 将链码执行的结果反序列化成order对象并应答前端

```
//查询订单列表
func orderList(ctx *gin.Context){3

}
```

7. 更新订单状态 思路：

1. 将前端提交的状态传入链码调用updateOrderStatus函数，当订单未完结时，依旧获取温度并上链，完结时，结束获取温度
2. 创建状态的请求体
3. 解析前端提交的数据并实例化对象
4. 创建一个用于状态枚举的map
5. 检查状态是否正确
6. 判断当前状态，运送中：设置线程自动查询温度，完成时：停止线程
7. 设置温度的查询时间以及查询的线程数组（方便以后停止线程时能找到）
8. 调用channelExecute执行updateOrderStatus，将对象的参数转成byte数组传入
9. 将链码执行的结果以json格式应答前端请求

```
// 更新订单温度列表
func updateOrderStatus(ctx *gin.Context){ 3

}
```

8. 查询账户列表 地址：perishable-food\application\controller\account.go 思路：
    1. 将前端提交的账户ID传入链码调用queryAccount函数
    2. 创建账户的请求体
    3. 解析前端提交的数据并实例化对象
    4. 调用channelQuery执行queryAccount，传入查询账户的ID（前端会传账户id或者all）
    5. 将链码执行的结果转成字符串并应答前端

```
// 查询账户列表
func accountList(ctx *gin.Context){ 3

}
```

任务3：行业痛点分析与解决方案编写

一、比赛要求

参赛选手根据现场下发的项目需求规格说明书中的题目，完善项目需求规格说明书。完成后将任务文件打包存放在“区块链竞赛/任务三”文件夹路径下。

二、比赛内容

（1）系统结构 根据题目绘制出结构图并完善系统结构说明 （2）主要业务功能 根据题目画出功能图 根据题目需求制定出供应商所拥有的功能 根据题目需求制定出买家所拥有的功能 根据题目需求制定出物流公司所拥有的功能 根据题目需求制定出智能合约所拥有的功能 （3）系统流程 根据题目需求绘制出系统流程图并说明

任务4：区块链理论考核

一、比赛要求

选手须根据给定场景，简答区块链结构及特性、运用区块链技术解决的行业痛点等方面知识，在word文档上做简答。

完成后将任务文件打包存放在“区块链竞赛/任务四”文件夹路径下

二、比赛内容

区块链结构及特性：什么是区块链，它有哪些特性 行业痛点分析：分析一下题目（易腐食品）解决了哪些行业痛点