Exercise – Blackboards

Following from the example pseudo code in the lecture slides, attempt to implement a simple blackboard container class for storing data of different types.

Your blackboard class could use a **std::map** to store entries based on an ID.

class Blackboard:

// returns true if id exists

def getEntry(id, value)

def setEntry(id, value)

def hasEntry(id)

def contains(id)

def getType(id)

// contains key-value pairs

entries

enum eBlackboardDataType:

type\_float,

type\_int,

type\_char,

type\_bool,  
 type\_pointer

struct BlackBoardData:

eBlackboardDataType type;

union:

float floatValue;

int intValue;

char charValue;

bool boolValue;

void\* pointerValue;

To be able to support the Expert / Arbiter model, implement the Expert and Arbiter classes from the lecture slides.

class Expert:

def getResponse()

def execute()

def arbiter()

bestExpert = None

bestResponse = 0

for expert in experts:

response = expert.getResponse()

if response > bestResponse

bestResponse = response

bestExpert = expert

if bestExpert:

bestExpert.execute()

class BlackboardQuestion:

experts

def arbiter()

Create a small test application to test that your implementation works as intended.

An example application might be a single Blackboard that a list of agents could access. One agent posts that it requires help. The remaining agents access the question and decide if they should respond, based on their own choice. Once all agents have responded the Arbiter decides which response was the best and executes the response. Perhaps the test application randomly places agents in an area and tags some as incapacitated, and the Arbiter makes its decision based on which Agent was closest to the one asking for assistance, where only agents that aren’t incapacitated respond.