Assignment #2 Real Deal FAQ

I don't know where to start.

Developing your solution incrementally will make your work easier. Start by making the trivial changes to the code so you can build the one source file having the name main.cpp. You don't have to solve the whole project, just get it to build without errors even if it doesn't work correctly. Then, starting with Robot, say, produce Robot.h, removing the code declaring the Robot class from main.cpp, but leaving in main.cpp the implementation of the Robot member functions. Get that two-file solution to build. Also, make sure you meet those of the requirements that involve only the Robot.h header.

Next, split off Player.h, testing the now three-file solution and also making sure you meet those of the requirements that involve only the Robot.h and Player.h headers. Continue in this manner until you've produced all the required headers, the whole program still builds, and you meet all the applicable requirements.

Now split off the member function implementations of, say, Robot, putting them in Robot.cpp. Build everything again. You see where this is going. Once you have the whole project building correctly in the eleven required files, then spend (the majority of) your time solving the project. The basic principle is to not try to produce all the files at once, because many misconceptions you have will affect many files. This will make it difficult to fix all those problems, since many of them will interfere with each other. By tackling one file at a time, and importantly, not proceeding to another until you've got everything so far working, you'll keep the job manageable, increasing the likelihood of completing the project successfully and, as a nice bonus, reducing the amount of time you spend on it.

I can't figure out what to pass as the first argument to the Robot constructor. I know it's supposed to be a pointer to the Arena the Robot is being added to.

Arena::addRobot needs to create a new Robot. Arena::addRobot is a member function of the Arena class. How does a member function of the Arena class talk about the Arena object it's operating on? If you answer this, you'll know what to do.

My program seems to work, but crashes when it quits. What's happening?

At the start of your Arena's destructor code, say

```
cerr << "Entering Arena destructor" << endl;
```

and at the end, say

```
cerr << "Leaving Arena destructor" << endl;
```

Is your destructor causing the crash? One possibility is that your destructor code itself is incorrect, but it's more likely that your destructor code is fine, but earlier in the program you were careless about managing your array of robot pointers, especially when a robot was destroyed.

How smart does takeComputerChosenTurn have to be?

Smart enough so that if staying put runs the risk of a robot possibly moving onto the player's location when the robots move, yet moving in a particular direction puts the player in a position that is safe when the robots move, then the chosen action is to move to a safer location. Similarly, if staying put is safe, but moving in certain directions puts the player in danger, then the chosen action should not be to move in one of the dangerous directions; instead, the player should stay put or move to another safe position. In general, a position that may be moved to by many robots is more dangerous than one that may be moved to by few. If the player doesn't move, then shooting at a robot, ideally the nearest one, is smart.

You don't have to be any smarter than that.

I'm still confused about how Arena::addRobot tells the Robot constructor what Arena the Robot is being added to.

Apologies to Abbott and Costello:

Teacher: FAQ #1 answers this.
Student: So what is the answer?
Teacher: This is the answer.
Student: What is?
Teacher: No, I told you this is the answer.
Student: What is?
Teacher: What is not.
Student: What is not what?
Teacher: What is not the answer.
Student: I don't care what is not the answer, I want to know what is the answer.
Teacher: But it's not.
Student: What's not?
Teacher: Right.
Student: Hunh? What's right?
Teacher: No, what's wrong.
Student: What's wrong is that you're not telling me the answer.
Teacher: But I did tell you the answer.
Student: Oh, this is absurd!
Teacher: This is not absurd. This is what you wanted to know.
Student: What is?
Teacher: This is.
Student: Look, you're not helping me by telling me that.

Teacher: I'm not telling you that.
Student: You're not telling me what?
Teacher: Not that, either.
Student: Not what?
Teacher: Right. I'm not telling you what.
Student: I don't want to play guessing games. Can't you just say something and tell me that's the answer?
Teacher: It would be lying to tell you that's the answer.
Student: It would be lying to tell me what's the answer?
Teacher: Right.
Student: Aw, the heck with it — I'll try and come up with this myself.
Teacher: Now you've got it!