CS 4318: Artificial Intelligence, spring 2022

Agent Challenge 5: Chocolate Russian roulette

In this agent challenge, you will write an agent function that plays a chocolate Russian roulette game against other agents. The game starts with a rectangular bar of chocolate divided into squares:

(0, 3)	(1, 3)	(2, 3)
(0, 2)	(1, 2)	(2, 2)
(0, 1)	(1, 1)	(2, 1)
(0, 0)	(1, 0)	(2, 0)

Players alternate taking bites from the upper right. The player that bites the (0, 0) square, which is poisonous, loses the game. If player A, going first, bites square (1, 2), the bar will look like

(0, 3)	(1, 3)	(2, 3)
(0, 2)	(1, 2)	(2, 2)
(0, 1)	(1, 1)	(2, 1)
(0, 0)	(1, 0)	(2, 0)

If player B then bites square (1, 0), the bar will look like

(0, 3)	(1, 3)	(2, 3)
(0, 2)	(1, 2)	(2, 2)
(0, 1)	(1, 1)	(2, 1)

(0, 0)	(1, 0)	(2, 0)

If player A then bites square (0, 1), the bar will look like

(0, 3)	(1, 3)	(2, 3)
(0, 2)	(1, 2)	(2, 2)
(0, 1)	(1, 1)	(2, 1)
(0, 0)	(1, 0)	(2, 0)

And then player B will be forced to eat the poisonous square, giving player A the win. In this example 3×4 game, player A ate a total of 7 chocolate squares and player B ate 5 (including the poisonous one).

Here's a Windows program you can use to practice playing the game: `ChocolateRR.exe`

Games will be played using chocolate bars ranging in size from 3×4 to 6×9. Using each bar size, each agent will play once as player A and once as player B against each other agent. The goals of each agent, in order of importance, are

- to win the largest number of games,
- to eat the largest number of chocolate squares (whether poisonous or not) and
- to take the smallest number of bites (whether poisonous or not).

Instructions:

1. Log in to `csunix.angelo.edu` and get a command prompt. Create a new directory named `crr` (maybe inside your `cs4318` directory).
2. Copy the file

   ```
   crr.zip
   ```

   to your `crr` directory.
3. Change to your `crr` directory and run

   ```
   unzip crr.zip
   ```

   at the command line. You will then have the following files:
   - `crr.h`: Contains `#includes` and definitions available to agent functions.
   - `crr.cpp`: Contains functions available to agent functions. The public member functions of the `ChocolateBar` and

`ChocolateSquareLocation` classes will be useful.
   - `crrMain.cpp`: Contains the `main` function that plays all agent functions against each other and summarizes the results.
   - `crrAgentBiteRight.cpp` and `crrAgentNibbleTop.cpp`: Contain agent functions for you to use as examples.
   - `crrAgentSmith.cpp`: Contains an agent function ready for you to make your own. This is the only file you should modify and turn in.
   - `crrBuild.bash`: A bash script that will build the executable file.
4. Run

```
nice bash crrBuild.bash
```

   at the command line to build the executable file. Then run

```
nice ./crrRunSim
```

   to play all given agents against each other and review the results.
5. Rename the file `crrAgentSmith.cpp` and the function inside it to reflect your agent name. For example, if your agent name were `Jones`, you would rename the `crrAgentSmith.cpp` file to `crrAgentJones.cpp` using the command

```
mv crrAgentSmith.cpp crrAgentJones.cpp
```

   and rename its `crrAgentSmith` function to `crrAgentJones`. Then fill your new agent function with your own agent code. Make sure that your code
   - uses no random-number generation,
   - uses no static or global variables,
   - has no other side effects (such as printing anything or saving to a file),
   - compiles without errors or warnings and
   - runs and finishes *quickly* without crashing.

   For testing purposes you may create as many agents as you like, each in its own separate code file, but you will turn in only one agent.
6. Once you have an agent function that you've tested thoroughly and are proud of, make sure your code is carefully commented. Then, in a comment below your agent function, describe in detail your agent's approach and why you expect it to do well against the other submitted agents. Also, if you talk about the assignment with anyone or give or receive any kind of help, document that activity in a comment.
7. Turn in your agent code file (such as `crrAgentJones.cpp`) using Blackboard by **11:00 p.m. on Monday, March 21st**. Do not turn in any other files. Official results will appear on this page soon after the agents are due.
8. Once the results for the first iteration appear below, analyze your agent's strengths and weaknesses, revise your agent code file and turn it in again using Blackboard by **11:00 p.m. on Wednesday, April 13th**. Take special care

to document in detail how and why you've improved your agent in light of its first-iteration performance. Further official results will appear on this page soon after the agents are due.

Unofficial results:

- Iteration 20220321 results
- Iteration 20220402 results
- Iteration 20220413 results

Official results:

- Iteration 1 results
- Iteration 2 results