REDPOINT
TECHNOLOGIES

Background and Requirements

The Chicago Blackhawks, being a tight-knit group, have traditionally bought end-of-the-season gifts for one another. For years, everyone on the team (including the head coach) has bought a little something for *everyone else on the team*. In recent years, with expanded rosters, heavier practice regimens and longer stays in the playoffs, they find they have less time to spend on shopping. Last season, they decided that each team member would buy a gift for *one other team member*, to be randomly selected (they drew pucks from a hat).

Thinking ahead to this holiday season, Kane and Toews mentioned that it would be great if everyone could have someone *new* to shop for, and that a random puck-from-a-bucket drawing would start to become more complicated if they had to eliminate last year's pairings. Coach Q, having a fair amount of gray matter north of his lower jaw (in addition to the ubiquitous gray 'stache), suggested that the Hawks' top-flight software engineering staff might be able to help them take the drudgery out of the giver/recipient matchmaking process.

On behalf of the mighty Blackhawks, we are requesting that you write a small program that meets the following requirements:

- Automatically and randomly pair team members as gift-givers and gift-receivers.
- Each team member must be both a gift-giver and a gift-receiver (there is no bench in this game).
- There should be no reciprocal pairings. In other words, if Duncan Keith is selected to buy for Brent Seabrook, then Seabrook should not be selected to buy for Keith.
- No pairing from last year's holiday season should be repeated this holiday season. If Kaner bought for Hossa last year, he gets to buy for someone new this year. (After all, how much fun is it to buy a size 54-long red sports jacket year after year?)

The Team	Last Year's Pairings
• Adam Burish	• Adam Burish bought for Duncan Keith
• Andrew Ladd	• Andrew Ladd bought for Joel Quenneville
• Antti Niemi	• Antti Niemi bought for Jonathan Toews
• Brent Seabrook	• Brent Seabrook bought for Kris Versteeg
• Bryan Bickell	• Bryan Bickell bought for Marian Hossa
• Brian Campbell	• Brian Campbell bought for Niklas Hjalmarsson
• Cristobal Huet	• Cristobal Huet bought for Patrick Kane
• Dave Bolland	• Dave Bolland bought for Patrick Sharp
• Duncan Keith	• Duncan Keith bought for Tomas Kopecky
• Joel Quenneville	• Joel Quenneville bought for Troy Brouwer
• Jonathan Toews	• Jonathan Toews bought for Adam Burish
• Kris Versteeg	• Kris Versteeg bought for Andrew Ladd
• Marian Hossa	• Marian Hossa bought for Antti Niemi
• Niklas Hjalmarsson	• Niklas Hjalmarsson bought for Brent Seabrook
• Patrick Kane	• Patrick Kane bought for Bryan Bickell
• Patrick Sharp	• Patrick Sharp bought for Brian Campbell
• Tomas Kopecky	• Tomas Kopecky bought for Cristobal Huet
• Troy Brouwer	• Troy Brouwer bought for Dave Bolland

You may write this program in any of the following languages that the Hawk's crack engineers have been studying (they will be doing ongoing maintenance). These languages include Java, C#, Visual Basic, Groovy and Ruby.

The Hawks' ethos under Coach Q. emphasizes straightforwardness and clear, simple communication – and this culture has trickled down to the supporting staff, including Software Engineering. They therefore prefer a solution that is concise, but not to the point of being cryptic.

This program should produce output in the following format**, with one line for each pairing:**
`<giverName> is buying for <receiverName>`

Your program, because it is assigning matches randomly, should produce different output each time it is run. (Of course, there is a slim chance that by random chance your program could generate the same results in back-to-back invocations, but the odds are *very* slim.

Sample Output

The sample below shows what the output from your program should look like. Note: these are the pairings from last year. Your pairings will be different, but the format should be as shown below.

```
Adam Burish is buying for for Duncan Keith
Andrew Ladd is buying for for Adam Burish
Antti Niemi is buying for for Joel Quenneville
Brent Seabrook is buying for for Kris Versteeg
Brian Campbell is buying for for Niklas Hjalmarsson
Bryan Bickell is buying for for Marian Hossa
Cristobal Huet is buying for for Patrick Kane
Dave Bolland is buying for for Patrick Sharp
Duncan Keith is buying for for Troy Brouwer
Joel Quenneville is buying for for Andrew Ladd
Jonathan Toews is buying for for Antti Niemi
Kris Versteeg is buying for for Tomas Kopecky
Marian Hossa is buying for for Brent Seabrook
Niklas Hjalmarsson is buying for for Bryan Bickell
Patrick Kane is buying for for Brian Campbell
Patrick Sharp is buying for for Cristobal Huet
Tomas Kopecky is buying for for Jonathan Toews
Troy Brouwer is buying for for Dave Bolland
```

Extra Credit

If you're as good with an algorithm as Hossa is with a hockey stick, this challenge might not have had you breaking a cerebral sweat. If there's still gas in your tank and you want to show us a little something extra, here's a requirement that was left off the immediate-need list (there's a year before it'll actually come into play). For a little additional challenge, you can implement this feature, too.

- The Hawks captains talked things over and decided that the whole "no-repeat" thing should extend beyond just the previous season. They feel that there shouldn't be any repeat giver-recipient pairs any time in the last four seasons. After four seasons, it's okay if a repeat gets assigned.
- To implement this feature, your program must be able to retain four seasons' worth of historical pairings (***in memory*** – no file or database persistence is required), and ensure that no pairing is repeated across the ***most recent four sets*** of assignments, beginning with last year's pairings provided above, and extending to those generated by your program.
- In order for the Hawks' QA department to be able to test your algorithm, they must be able to have it generate more than one set of pairings per execution. To allow for this to occur, your program should accept console input. If the user types a 'Q' or 'q' and hits Enter, the program should quit (exit to the OS). Anything else (e.g. blank line) should generate and display a new set of pairings.

Submitting Your Solution

If you spend the time and energy solving this problem, we would ***love*** to take a look at your solution! Here's all you need to do:

- Package your solution into a single file (jar, zip, tar, etc.). If you used a common IDE (Visual Studio, Eclipse, etc.), go ahead and use the IDE to generate the archive; having access to your project metadata makes it easier for us to load 'er up and go.
- Send your solution as an attachment to challenge@redpointtech.com.
- If you are doing this purely for fun and do ***not*** want us to contact you regarding possible employment, just say so in your email, and we'll certainly respect your wishes. Either way, we'll take a look at your code and let you know what we think!

That's it! Happy problem solving...