

WES-CS GROUP MEETING #1

Exercise 1: Team Robot

A common pattern for computer programs is to get input and process it to generate some desired output. Many *systems* can also be modelled in this manner, including robots. A simple robot has sensors that gather input. The input is passed to a controller that processes it and determines what output to do. The output directs manipulators to change the robot's environment to achieve some desired goal.

For this exercise you will work with a team to simulate the parts of a robot. Your robot team will be asked to arrange items into specific goal configurations. Your robot team is divided into the following parts:

- **The Brain** (processor/controller). One member of the team plays the *brain*. Only the brain can see the goal configuration. The brain can ask simple questions of the *eyes* (questions to which only short answers are needed). The brain can also tell the *hands* what to do and again should try to keep the commands simple. The brain cannot see the work area where the hands are arranging the items or ask the hands any questions.
- **The Eyes** (input/sensors). One member of the team plays the *eyes*. Only the eyes can see the work area. The eyes can only respond to the brain's questions and should try to keep the answers short and simple. The eyes cannot see the goal configuration. The eyes cannot ask any questions or speak to the hands.
- **The Hands** (output/manipulators). Two members of the team play the *hands*. One plays the left hand and the other plays the right hand. (If you have only three members on your team one person can play both hands.) Only the hands can move items in the work area. The hands respond to the brain's commands. The hands cannot see or speak (but can listen to the brain).

Part (a). Divide into groups of 3 or 4. Decide who will be playing which part. Make sure the brains can't see the work area, and that the hands have put on their blindfolds. Your Team Leader will now show the brains a picture of the goal configuration to be built from the items in the work area, and the eyes will empty the contents of the bag onto the work area. Your team should try to make the goal configuration as quickly and accurately as possible. After a certain amount of time your Team Leader will say stop and reveal the goal configuration to everyone.

Part (b). Have the members on your team try playing different parts of the robot to build another (different) goal configuration.

For Discussion:

1. Which of the parts was hardest to play? Why? Which was easiest? Why?
2. What type(s) of item configurations were hardest or easiest to arrange? Why?
3. What kind(s) of questions/commands were most useful? Why?
4. What kinds of behaviors of team members were helpful? Not so helpful (be nice)?

Part (c) (optional). Have a competition among teams to see who can reach the goal most quickly and accurately. Develop a scoring system to judge the results.

Exercise 2: Working in Groups

The robot exercise required everyone to work together cooperatively. In general, being able to work well in a group is an important skill. For this exercise, look at the descriptions of what some people do when they work in a group. Which describe attributes that you'd like people in your group to have? Which would you rather avoid for people in your group?

Add some descriptions of your own to each category.

Finally, choose at least one positive and one negative attribute that you think apply to you.

Exercise 3: Logical Thinking (Sudoku)

One of the benefits many students find they get from taking Computer Science courses is that it helps develop their logical-thinking skills. We'll work on that today by trying some Sudoku puzzles. Here's a simplified example:

1			2
	3	2	

The whole 4-by-4 thing is called a *grid*; each 2-by-2 piece is called a *box*; and the 16 individual parts are called *squares*.

The object is to fill in the empty squares with numbers from 1 to 4 so that the same number doesn't appear twice in any **row**, or any **column**, or any **box**.

Part (a). Work with a partner to solve the puzzle given above. To solve the puzzle you could just guess, but that won't work very well. Instead, see if you can find one square that's *forced* to have a particular number in it. Wait until both you and your partner have found a square like that, then explain to each other which square it is, and why you think it needs to have the number you put in there. If you think your partner made a mistake, explain why (nicely!). Then work together to solve the whole puzzle.

Part (b). Now try solving a real Sudoku puzzle (9-by-9 instead of 4-by-4). Hint: start by looking at a row, column, or box with a lot of numbers already in it.

5		6						
	2			8		9	7	1
	8				4		3	
2			8			7		
			7		1			
		8			5			9
	6		1				9	
1	4	9		5			2	
						3		5

Part (c).

Before we try any more Sudoku puzzles, let's think more about a systematic way to solve them. If you wanted to write a Java program to solve Sudoku puzzles, you'd have to come up with some rules for how to fill in the squares. One way to do that is as follows:

- First, fill in every blank square with a list of all of the numbers that might go there: the numbers that don't appear in the same row, column, or box.
- If you filled in any of the squares with a single number, then that's the final answer for that square, and you can use it to eliminate that number from the lists in the squares in the same row, column, and box. That may cause some other list to turn into a single number, and then you can repeat the process over and over.

This technique works for a lot of Sudoku puzzles (though it's kind of tedious to do it by hand). First try it on our small puzzle:

1			2
	3	2	

Work with a partner to solve this puzzle using the rules given above. Then compare your solution with the other pairs in the class.

The technique you just used won't solve all puzzles. Try this one (note that this puzzle has more than one solution):

2			
4		2	
		4	

See if you can figure out any rules for eliminating numbers from some lists. If you can eliminate enough to make a list turn into a single number, then you can go back to the technique we've been trying. There are several interesting rules that you might be able to discover for this puzzle. Once you've done that, go ahead and try to solve some more full-sized puzzles (on the last two pages).

			1			7		2
	3		9	5				
		1			2			3
5	9					3		1
	2						7	
7		3					9	8
8			2			1		
				8	5		6	
6		5			9			

9	4					3	1
	8		9		6	2	
2		6				8	9
			1		3		
7							5
			8		2		
1		2				3	8
	7		3		1	5	
4	5					9	7