

WES-CS GROUP MEETING #2

Exercise 1: Jelly Sandwich (Algorithms)

Remember that computers are fast but dumb. They will do exactly what they're told, even if it's the wrong thing. They can only follow simple, step-by-step instructions. A list of such instructions is called an *algorithm*.

For this exercise, we'll explore algorithms for making a jelly sandwich. Assume you are writing a list of instructions (in English) to make a jelly sandwich for your new friend who is not so smart.

Part (a). Work with a partner to come up with an algorithm for making a jelly sandwich. Write it down and give it to your Team Leader.

Part (b). Try writing a new algorithm based on what just happened.

For Discussion:

1. What does this exercise reveal about the English language?
2. What is needed to be able to write a precise algorithm?
3. Are there any advantages to acting out a proposed algorithm?

Exercise 2: Picasso (Java Programming)

A Java program involves creating and manipulating *objects*, each of which provides some *operations*. An operation can either perform a task (like printing something on the computer screen), or it can do a computation and tell you the answer. Some operations require that you provide values to be used in their task/computation.

For this exercise, we'll assume that we have an *Artist* object named *picasso* that draws in a two dimensional grid of cells. It provides the following operations, each of which performs a task:

`drawLineDown(int length)`

Draw a vertical line starting from the current position and going straight down for the given number of cells. The current position is changed to be at the bottom end of the line.

`drawLineUp(int length)`

Draw a vertical line starting from the current position and going straight up for the given number of cells. The current position is changed to be at the top end of the line.

`drawLineRight(int length)`

Draw a horizontal line starting from the current position and going straight to the right for the given number of cells. The current position is changed to be at the right end of the line.

`drawLineLeft(int length)`

Draw a horizontal line starting from the current position and going straight to the left for the given number of cells. The current position is changed to be at the left end of the line.

`moveRight(int d)`

Move the current position *d* cells to the right.

`moveLeft(int d)`

Move the current position *d* cells to the left.

`moveUp(int d)`

Move the current position *d* cells up.

`moveDown(int d)`

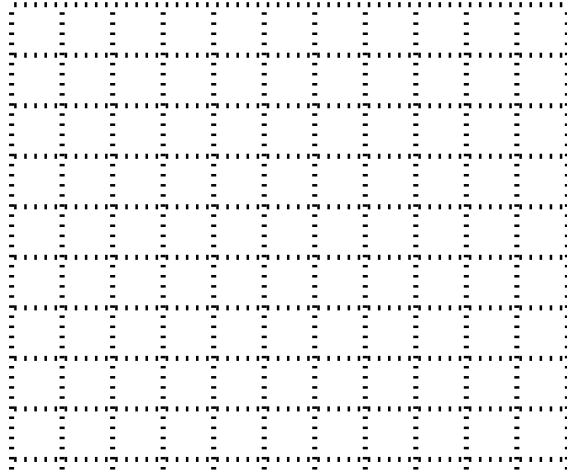
Move the current position *d* cells down.

Part (a). What is drawn when the following code executes? (Use the grid to do the drawing; assume that the current position starts in the top left corner of the grid.)

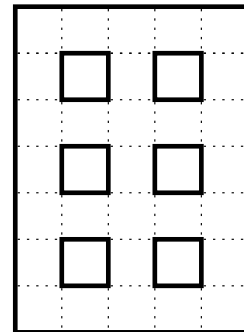
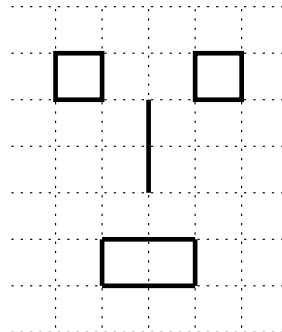
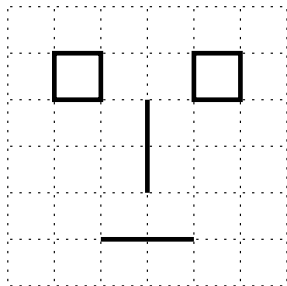
```

picasso.drawLineDown( 7 );
picasso.moveUp( 4 );
picasso.drawLineRight( 2 );
picasso.moveUp( 3 );
picasso.drawLineDown( 7 );
picasso.moveRight( 2 );
picasso.drawLineRight( 2 );
picasso.moveLeft( 1 );
picasso.drawLineUp( 7 );
picasso.moveLeft( 1 );
picasso.drawLineRight( 2 );

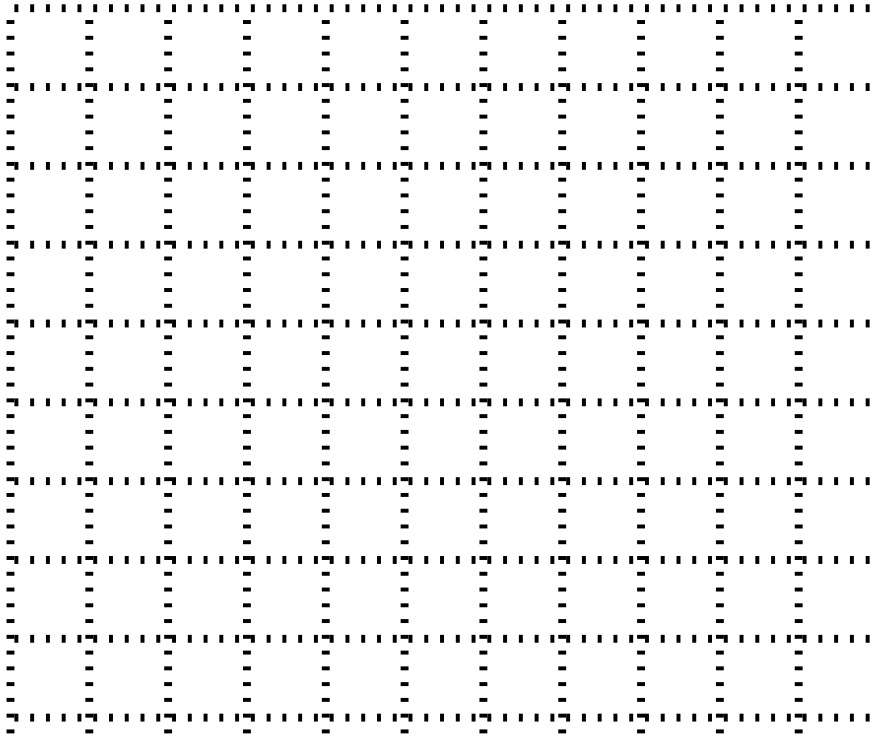
```



Part (b). Divide into groups of two. Each group choose one of the pictures shown below and write Java code that would make *picasso* draw the picture. Assume that the current position starts in the top left corner of the grid. (The dotted lines are not part of the pictures; they just show you the cells in the grids.)



Part (c). Now divide into three groups, each using one laptop. In the grid below, draw your own picture and then write Java code for just a small part of your drawing. Enter the code on the laptop and run the program to see the result. Once you get a part of your drawing working, do another part, then another and another until you've completed the program for your drawing. This is called *incremental development*.



For discussion:

1. What does this exercise reveal about the Java language?
2. What other *Artist* methods would have made it easier to draw the pictures in Part (b)?
3. For Part (a), you traced out by hand what a piece of code would do. Does this seem like it might be useful when you write Java programs? Why or why not?
4. Is there only one right answer for Part (b)? What makes one answer better than another?
5. Most programming languages, including Java, let you write *loops*. For example, in Java you can essentially say “repeat the following commands *n* times”. Can any of the code for the pictures in part (b) be simplified by using loops?
6. Why do you think *incremental development*, like you used in Part (c), might be useful when you write Java programs?

Exercise 3: Twisted Words (Algorithms)

For this exercise, we'll combine the idea of *algorithms* from the jelly-sandwich exercise, and the idea of writing simple Java code, like in the *picasso* exercise.

This time, instead of an `Artist` called `picasso`, we'll assume that we have something called `word`, which represents an English word. The `word` has operations that rearrange the letters in the word and an operation that prints the word. Most of the operations have one or two integer arguments that specify *positions* in the word. Because Java (and many other modern programming languages) starts counting from zero, we'll do that for the positions, too. For example, if `word` represents `CANDY`, we'll say that the letter `C` is in position zero, the letter `A` is in position one, and so on.

Below are descriptions of the operations.

```
moveToFront( int pos )
```

Move the letter in position `pos` to the beginning of the word (i.e., to position zero).

```
moveToEnd( int pos )
```

Move the letter in position `pos` to the end of the word.

```
swap( int pos1, int pos2 )
```

Swap the letter in position `pos1` with the letter in position `pos2`.

```
print( )
```

Print the word.

Part (a). Below are some examples. In each case, the letter or letters that will be moved are shown in bold in the first column. Work with a partner to write the results in the third column, then compare your answers with another pair.

Original word	Java code	word after the code executes
POST	<code>word.moveToFront(2)</code>	
STOP	<code>word.moveToEnd(0)</code>	
TOPS	<code>word.swap(0, 2)</code>	

Nothing on this page!

Part (b).

Remember that an algorithm is a clear, step-by-step description of how to perform a task. Below is an algorithm that does something to a word.

Step 1: Let j be 0 and let k be the number of letters in the word $- 1$.

Step 2: while ($j < k$) repeat the following two sub-steps.

Sub-step (i): `word.swap(j, k)`

Sub-step (ii): Let j be $j + 1$ and let k be $k - 1$.

Step 3: `word.print()`

What is the result of executing this algorithm if `word` is initially `PUPILS`?

What if `word` were initially `HELLO`?

Tracing an algorithm on a specific example can help us figure out what the algorithm does in general. What does the algorithm above do in general?

Part (c).

Now assume we only have the `moveToFront` method. Write an algorithm that has the same effect on the word as the algorithm given above.

If you finish early, try it again assuming that you only have the `moveToEnd` method.

Exercise 4: Age and Shake (Logical Thinking)

Here are two logic puzzles. Try solving them now. If you don't finish, you can keep working on them during the week if you want to. We'll talk about the answers next time.

1. What are the ages?

Two mathematicians are sitting together in a building. They don't have anything to do at the moment, so one says to the other, "*Try guessing the ages of my three children. I'll give you a hint - the product of their ages is 72.*" The other mathematician says, "*That's not enough information. Tell me more.*" The first mathematician says, "*Their ages add to be the number of this building.*" The other mathematician goes outside, looks at the building number, comes back, and says, "*That still isn't enough information. Tell me more.*" The mathematician says, "*My youngest child's name is Anne.*"

Problem: What are the ages of the children?

2. How many handshakes?

A woman and her partner attend a party with four other couples. Some people shake hands with other people. Of course, no one shakes their own hand or the hand of the person they came with. When the woman asks the other (9) people present how many different people's hands they shook they all gave a different answer.

Problem: How many different people's hands did the woman's partner shake?