CS0411 INTRODUCTION TO CS PROGRAMMING

Promotion, Demotion and Casting

Primitive	Valid promotion
`double`	`None`
`float`	`double`
`long`	`float, double`
`int`	`long, float, double`
`short`	`int, long, float, double`
`byte`	`short, int, long, float, double`
`char`	`int, long, float, double`
`boolean`	`none`

Promotion may be **implicit**, for example **`double d = i;`**
or **explicit** by using a casting operator, for example **`double d = (double)i1/i2;`**

Java will implicitly promote when there is a mixture of types in a calculation, for example **`double d2 = d1/i1`**; will do regular division (**`i1`** is implicitly promoted to a **`double`**).

Demotion can never be **implicit**, **`int i = 3.7;`** is a compile error
You must explicitly use the casting operator for demotion. When demoting there may be a loss of accuracy. The value of **`(int)3.7`** is **3**. The value of **`(int)3.0`** is **3**.

The casting operator has a higher precedence than mathematical operators (+, -, *, /, %)
Casting does not change the actual variable value that the cast is done to.

EXAMPLE: Given the starting code:

```
double d1, d2, d3;
int i1, i2, i3;
i1 = 2;
i2 = 5;
d1 = 5.6;
d2 = 4.0;
```

decide whether each of the following would compile. **For those that do, write down the result of the line. For those that do not, explain why not**. Treat each line separately as following on from the 6 lines above:

#1	`i3 = 2;`	
#2	`i3 = i2;`	
#3	`i3 = d1;`	
#4	`i3 = d2;`	

#5	`i3 = (int)d1;`	
#6	`i3 = (int)d2;`	
#7	`i3 = d1/i1;`	
#8	`i3 = (int)d1/i1;`	
#9	`d3 = (int)d1/i1;`	
#10	`i3 = d2/i1;`	
#11	`i3 = (int)d2/i1;`	
#12	`d3 = i1;`	
#13	`d3 = i1*d1;`	
#14	`d3 = i1*d2;`	
#15	`d3 = i2/i1;`	
#16	`d3 = (double) i2/i1;`	
#17	`d3 = (double) (i2/i1);`	