Question 1:

```
public class BlankKarel extends SuperKarel {
	public void run() {
		while (frontIsClear()) {
			walkARow();
			move();
		}
		walkARow();  // fencepost
	}

	public void walkARow() {
		turnLeft();
		move();
		while(frontIsClear() && leftIsBlocked()
				&& rightIsBlocked()) {
			move();
		}
		turnAround();
		if (leftIsBlocked() && rightIsBlocked()) {
			placeBeepers();
		} else {
			returnHome();
		}
	}

	public void placeBeepers() {
		while (frontIsClear()) {
			putBeeper();
			move();
		}
		turnLeft();
	}

	private void returnHome() {
		while (frontIsClear()) {
			move();
		}
		turnLeft();
	}
}
```

Students generally solve this in one of two ways. Either they move up the column until they hit a wall or a break on the left or right, and beeper on the way down if it is called for, or they beeper on the way up and pick up the beepers on the way down if necessary.

Question 2:

Part a:

Answer:

9

8

Part b: Result Table

12345
22345
33345
44445
55555

Part c: Result Table

In unknown, num1 is 7
In unknown, num2 is -3
In unknown, num3 is 1
In mystery, num2 is 1
In unknown, num1 is 7
In unknown, num2 is -6
In unknown, num3 is -5
In mystery, num2 is 1
In mystery, num4 is -5
3

Problem 3

```
public class Item {
      // what instance variables do you need?
      private int dailySales;
      private double pricePaid;
      private double priceSold;
      private int itemQuantity;
      private String itemName;

      // Note that the wholeSale price is what the store must pay to purchase
      // the item while the sellingPrice is what the store sells the item for
      public Item (String name, double wholesalePrice, double sellingPrice) {
            itemName = name;
            pricePaid = wholesalePrice;
            priceSold = sellingPrice;
            itemQuantity = 0;
            dailySales = 0;
      }

      public Item (String name, double wholesalePrice, double  sellingPrice,
            double numberInStock)  {
            itemName = name;
            pricePaid = wholesalePrice;
            priceSold = sellingPrice;
            itemQuantity = numberInStock;
            dailySales = 0;
      }

      public void addInventory(int numberToBeAdded)  {
            itemQuantity += numberToBeAdded;
      }

      // the next method returns true/false depending on whether there are
      // enough items in the inventory to sell (if there aren't, no sale
      //should happen
      public boolean sell(int quantity)  {
            if (quantity <= itemQuantity) {
                  itemQuantity -= quantity;
                  dailySales += quantity;
                  return true;
            } else {
                  return false;
            }
      }
```

```
	// should return a String contianing the item's name, and the
	// quantity in stock
	public String toString() {
		return itemName + " " + itemQuantity;
	}

	// should return the money made from the daily sales of the item
	// note that the profit is the sellingPrice minus the wholesalePrice
	public double dailyProfit() {
		return dailySales * (priceSold – pricePaid);
	}

	// resets the number of items sold to 0
	public void resetSales()  {
		dailySales = 0;
	}
```

}

Question 4:

```
//
// preconditions: num > 1 and (num-1) < CAKE_WIDTH / 2 (or, can't have so
//     many candles that they can't fit on the cake)
//
private void addCandles (int x, int y, int num) {
	if (num > 1 && (num – 1) < (CAKE_WIDTH / 2)) {
		int offset = CAKE_WIDTH/(num - 1);
		for (int i = 0; i< num;i++) {
			GLine line = new GLine (x+i*offset,y,
					x+i*offset,y-CANDLE_HEIGHT);
			add(line);
		}
	}
}
```

Question 5:

```
// using indexOf
private String longest (String str) {
      String result = "";
      int pos;
      do {
            pos = str.indexOf(" ");
            if (pos != -1) {
                  if (result.length() < pos)
                        result = str.substring(0,pos);
                  str = str.substring(pos+1);
            } else if (result.length() < str.length())
                  result = str;
      } while (pos != -1);

      return result;

}

// using StringTokenizer
private String longest2 (String str) {
      String result = "";
      StringTokenizer st = new StringTokenizer(str);
      while (st.hasMoreTokens()) {
            String word = st.nextToken();
            if (word.length() > result.length())
                  result = word;
      }
      return result;
}


// using charAt
private String longest3 (String str) {
      String result = "";
      int pos;
      do {
            pos = 0;
            while (pos < str.length() && str.charAt(pos)!=' ')
                  pos++;

            if (pos < str.length()) {
                  if (result.length() < pos)
                        result = str.substring(0,pos);
                  str = str.substring(pos+1);
            } else if (result.length() < str.length())
                  result = str;

      } while (pos < str.length());
      return result;
}
```

```
// using charAt, not breaking apart the input string
private String longest4 (String str) {
	String result = "";
	int pos = 0;
	String temp = "";
	for (int i=0; i< str.length();i++) {
		if (str.charAt(i) == ' ') {
			if (temp.length() > result.length())
				result = temp;
			temp = "";
		} else {
			temp += str.charAt(i);
		}
	}
	// handle last word
	if (result.length() < temp.length())
				result = temp;
	return result;
}
```