```java
// James Van Gilder
// 9081186117

public class Main{
   // Q A
   public int add(int x, int y){
       // ensure that neither x nor y are greater than the max val
       if ((x > Integer.MAX_VALUE) || (y > Integer.MAX_VALUE)){
           throw new ArithmeticException("Value too large!");
       }
       // ensure that neither x nor y are less than the min val
       else if ((x < Integer.MIN_VALUE) || (y < Integer.MIN_VALUE)){
           throw new ArithmeticException("Value too small!");
       }
       int result = x + y;
       // ensure that the resulting sum is not greater than the max val
       if (result > Integer.MAX_VALUE){
           throw new ArithmeticException("Sum too large!");
       }
       // ensure that the resulting sum is not less than the min val
       else if (result < Integer.MIN_VALUE){
           throw new ArithmeticException("Sum too small!");
       }
       // ensure that if both x and y are positive, the sum is positive
       // to avoid wrap around
       else if (((x >= 0) && (y >= 0)) && result < 0){
           throw new ArithmeticException("Sum too large!");
       }
       // ensure that if both x and y are negative, the sum is negative
       // to avoid wrap around
       else if (((x <= 0) && (y <= 0)) && result > 0){
           throw new ArithmeticException("Sum too small!");
       }
       // ensure that if x is negative and y is positive
       // the resulting sum is less than y and greater than x
       // to avoid wrap around
       else if (((x <= 0) && (y >= 0)) && (result > y || result < x)){
           throw new ArithmeticException("Sum too large!");
       }
       // ensure that if x is positive and y is negative
       // the resulting sum is less than x and greater than y
       // to avoid wrap around
       else if (((x >= 0) && (y <= 0)) && (result < y || result > x)){
           throw new ArithmeticException("Sum too large!");
       }
       return(result);
   }
   // Q B
   public int subtract(int x, int y){
        // ensure that neither x nor y are larger than the max val
        if ((x > Integer.MAX_VALUE) || (y > Integer.MAX_VALUE)){
            throw new ArithmeticException("Value too large!");
        }
        // ensure that neither x nor y are less than the min val
        else if ((x < Integer.MIN_VALUE) || (y < Integer.MIN_VALUE)){
            throw new ArithmeticException("Value too small!");
        }
        int result = x - y;
        // ensure that the resulting difference is not greater than the max val
        if (result > Integer.MAX_VALUE){
            throw new ArithmeticException("Difference too large!");
        }
        // ensure that the resulting difference is not less than the min val
        else if (result < Integer.MIN_VALUE){
            throw new ArithmeticException("Difference too small!");
        }
        // ensure that if both x and y are positive, the resulting
        // difference is not greater than either value to avoid wrap around
        else if (((x >= 0) && (y >= 0)) && ((result > x) || (result > y))){
            throw new ArithmeticException("Difference too small!");
        }
        // ensure that if both x and y are negative, the resulting
        // difference is not less than either value to avoid wrap around
        else if (((x <= 0) && (y <= 0)) && ((result < x) || (result < y))){
            throw new ArithmeticException("Difference too large!");
        }
        // ensure that if x is positive and y is negative the resulting
        // difference is not less than x or y to avoid wrap around
        else if (((x >= 0) && (y <= 0)) && ((result < x) || (result < y))){
            throw new ArithmeticException("Difference too large!");
        }
        // ensure that if x is negative and y is positive the resulting
        // difference is not greater than x or y to avoid wrap around
        else if (((x <= 0) && (y >= 0)) && ((result > x) || (result > y))){
            throw new ArithmeticException("Difference too large!");
        }
        return(result);
    }
    // Q C
    public int multiply(int x, int y) {
        // ensure that neither x nor y are greater than the max val
        if ((x > Integer.MAX_VALUE) || (y > Integer.MAX_VALUE)) {
            throw new ArithmeticException("Value too large!");
        }
        // ensure that neither x nor y are less than the min val
        if ((x < Integer.MIN_VALUE) || (y < Integer.MIN_VALUE)) {
            throw new ArithmeticException("Value too small!");
        }
```

```
int result = 0;
if (y == 0) {
    return (result);
}
else if (y > 0) {
    while (y > 0) {
        result += x;
        if (result > Integer.MAX_VALUE) {
            throw new ArithmeticException("Product too large!");
        }
        if (result < Integer.MIN_VALUE) {
            throw new ArithmeticException("Product too small!");
        }
        y--;
    }
}
else if (y < 0) {
    while (y < 0) {
        result -= x;
        if (result > Integer.MAX_VALUE) {
            throw new ArithmeticException("Product too large!");
        }
        if (result < Integer.MIN_VALUE) {
            throw new ArithmeticException("Product too small!");
        }
        y++;
    }
}
// ensure that if x and y are both positive, the product is too
// to avoid wrap around
if (((y > 0) && (x > 0)) && result <= 0) {
        throw new ArithmeticException("Sum too small!");
}
// ensure that if y is negative and x is positive, the product
// is positive to avoid wrap around
else if (((y < 0) && (x > 0)) && result >= 0) {
    throw new ArithmeticException("Sum too large!");
}
// ensure that if y and x are both negative, the product
// is positive to avoid wrap around
else if (((y < 0) && (x < 0)) && result <= 0) {
    throw new ArithmeticException("Sum too large!");
}
// ensure that if y is positive and x is negative, the product
// is positive to avoid wrap around
else if (((y > 0) && (x < 0)) && result >= 0) {
    throw new ArithmeticException("Sum too small!");
}
else return(result);
```

```
}
// Q D
public int divide(int x, int y) {
    // ensure that there are no divide by zero errors
    if (y == 0){
        throw new ArithmeticException("divide by zero error");
    }
    // ensure that neither x nor y are greater than the max val
    if ((x > Integer.MAX_VALUE) || (y > Integer.MAX_VALUE)) {
        throw new ArithmeticException("Value too large!");
    }
    // ensure that neither x nor y are less than the min val
    if ((x < Integer.MIN_VALUE) || (y < Integer.MIN_VALUE)) {
        throw new ArithmeticException("Value too small!");
    }
    // ensure that if x = 0, zero is returned
    if (x == 0){
        return 0;
    }
    int result = x / y;
    // ensure that the quotient is not greater than the max val
    if (result > Integer.MAX_VALUE) {
        throw new ArithmeticException("Quotient too large!");
    }
    // ensure that the quotient is not less than the max val
    else if (result < Integer.MIN_VALUE) {
        throw new ArithmeticException("Quotient too small!");
    }
    // ensure that if both x and y are positive that the quotient
    // is also positive to avoid wrap around
    else if (((x > 0) && (y > 0)) && result <= 0) {
        throw new ArithmeticException("Quotient too large!");
    }
    // ensure that if x is negative and y is positive, the
    // quotient is negative to avoid wrap around
    else if (((x < 0) && (y > 0)) && result >= 0) {
        throw new ArithmeticException("Quotient too small!");
    }
    // ensure that if x and y are both negative that the quotient is
    // also negative to avoid wrap around
    else if (((x < 0) && (y < 0)) && result <= 0) {
        throw new ArithmeticException("Quotient too large!");
    }
    // ensure that if x is positive and y is negative, the
    // quotient is negative to avoid wrap around
    else if (((x > 0) && (y < 0)) && result <= 0) {
        throw new ArithmeticException("Quotient too large!");
    }
    return result;
```

```
        }
}
```