CSE 331 Section 6 Midterm Review Worksheet

1. Fill in the implementation of a method that converts a **positive integer** to its **string representation in decimal**. Useful facts to remember:
   a. Convert **`char ch`** that is one of **`'0'`**, **`'1'`**, …, **`'9'`** to a corresponding int by doing **`ch - '0'`**
   b. Convert **`int x`** that is one of **`0`**, **`1`**, …, **`9`** to a corresponding char by doing **`(char) (x + '0')`**

```
{{ P: x > 0 }}
String intToString(int x) {
  StringBuilder buf =
  int k =  , y =  ;
  {{ Inv: P and buf stores the lowest k digits of x
          in reverse order and y = x / 10^k }}
  while (y != 0) {









   k = k + 1;
  }

  return buf.reverse().toString();
}
```

2. Consider the following three method specifications.
   A. **`@effects decreases balance by amount`**
   B. **`@requires amount >= 0 and amount <= balance`**
      `@effects decreases balance by amount`
   C. **`@throws InsufficientFundsException if balance < amount`**
      `@effects decreases balance by amount`

   Which specifications does each of the following four implementations satisfy?

```
void withdraw(int amount) {
   balance -= amount;
}

void withdraw(int amount) {
  if (balance >= amount) balance-=amount;
}
```

```
void withdraw(int amount) {
  if (amount < 0) throw new IllegalArgumentException();
  balance -= amount;
}

void withdraw(int amount) throws InsufficientFundsException {
  if (balance < amount) throw new InsufficientFundsException();
  balance -= amount;
}
```

3. Consider the **BankAccount** class. What are some good test cases?

```
public class BankAccount {
  /** @return current balance of account */
  public void balance() { … }

  /**
   * @param amount to withdraw
   * @requires amount >= 0
   * @throws InsufficientFundsException
   *         if balance < amount
   * @effects decreases balance by amount
   */
  public void withdraw(int amount) { … }
}
```

4. Verify that the following method is correct:

```
/** Return the value of this IntPoly at point x */
public int valueAt(int x) {
  int val = a[0];
  int xk = 1;
  int k = 0;
  int n = a.length - 1;
  // 4.1
  {{ inv: xk = x^k && val = a[0] + a[1]*x + … + a[k]*x^k }}
  while (k != n) {
    {{_____________________________________________________}}
    xk = xk * x;
    {{_____________________________________________________}}
    val = val + a[k+1]*xk;
    {{_____________________________________________________}}
    k = k + 1;
    {{_____________________________________________________}}
  }
  // 4.2
  {{ val = a[0] + a[1]*x + … + a[n]*x^n }}
  return val;
}
```

4.1. Why does the invariant hold before the loop?

4.2. Why does the postcondition hold after the loop?