Testing the Edges Activity – Test Implementation Scenarios Handout

Problem Statement

We need a function to round a number (integer) to the nearest multiple of 10. For example, 24 should be rounded to 20, and -27 should be rounded to -30. A number which ends in 5, such as 15 or -15, should be rounded "away from zero," such that 15 rounds to 20 and -15 rounds to -20.

Implementation I:

This implementation (tries to) get the last digit of the number in lastDigit (using the % modulo operator) and the number without its last digit (by dividing by 10, truncating it). It then determines whether to add 10 to the truncated number based on whether the last digit is 5 or greater. It fails for negative numbers because the % operator returns a negative result when its first operand is negative, so it will never find that a negative number has a last digit that is 5 or greater.

```
public static int round(int n)
{
    int noLastDigit = n / 10;
    int lastDigit = n % 10;
    if (lastDigit < 5)
        return noLastDigit * 10;
    else
        return noLastDigit * 10 + 10;
}
```

Implementation II:

This implementation (a) adds five to the number (changing 17 to 22, for example), then divides by 10 with the result always being an integer (so 22 / 10 = 2), then multiplies that result by 10 to undo the division, with the fractional portion of the number having been discarded by the integer division (so 2 * 10 = 20). It fails because adding 5 to -17 produces -12, which truncates to -1 and then multiplies to -10 instead of -20.

```
public static int round(int n)
{
    return (n + 5) / 10 * 10;
}
```

Implementation III:

This implementation is like implementation II, except that it subtracts 5instead of adding 5 when the parameter is negative. So -17 will become -22, get truncated to -2, then multipled to -20. This implementation is correct.

```
public static int round(int n)
{
    if (n < 0)
        return (n - 5) / 10 * 10;
    else
        return (n + 5) / 10 * 10;
}
```