

Programming I

Lab 6

Exercise 1

Without using a Java compiler, trace through the following code segments to determine what the output will be.

Segment 1

```
int i = 7;
int j = 9;
while ((i > 0) && (j > 0))
{
    i--;
    j--;
    System.out.print(i + " ");
}
output: _____
```

Segment 2

```
int i = 7;
int j = 9;
while ((i > 0) || (j > 0))
{
    i--;
    j--;
    System.out.print(i + " " + j + " ");
}
output: _____
```

Segment 3

```
int i = 13;
int j = 7;
while ((i > 0) && (j != 0))
{
    i = i-2;
    j--;
    System.out.print(i + " " + j + " ");
}
output: _____
```

Segment 4

```
for (int i = 0; i < 10; i++)
{
    if (i % 3 == 0)
        System.out.print(i + " " + i + " ");
    else
        System.out.print(i + " ");
}
output: _____
```

Segment 5

```
for (int i = 0; i < 10; i++)
{
    if (i == 0 || i == 9)
        System.out.print("$");
    else if (i > 3 && i < 7)
        System.out.print("%");
    else
        System.out.print("#");
    System.out.print("@");
}
```

output: _____

Exercise 2

Consider the following code segment:

```
for (int i = 1; i < 5; i++)
{
    for (int j = 0; j <= i; j++)
    {
        System.out.print("* ");
    }
    System.out.println();
}
```

Trace the code (see Unit 6) showing how the variables `i` and `j` change with every loop iteration as well as the corresponding output. How would you change the code segment so that it produces the following output?

```
* * * *
* * *
* *
*
```

Exercise 3

Express each the following set of alternatives as a conditional statement (using `if/else` or `if/else-if` structures) pertaining to **num**:

1. a) num is divisible by 3, b) all other values
2. a) num is in the range 5-10 or num is odd, b) all other values
3. a) num is in the range 0-999 but not 333, 444, or 555 b) num is greater than 999, c) all other values

Programming exercise 1

Part A

Using a nested **while loop** structure (a loop inside a loop), write a program that animates a red cell moving vertically down rows in a 20x15 board object. The cell starts at the top left-hand

corner and ends at the bottom right hand corner. When the cell reaches the last row of a column it moves to the first row of the next column.

Part B

Modify your program from Part A in the following 3 ways:

- The program should use a for loop instead of a while loop.
- Instead of creating a 20x15 board, prompt the user to enter its dimensions and create the board accordingly.
- Add a green cell that starts at the bottom right hand corner of the board and moves vertically up from bottom to top (the reverse of what the red cell does). The movement of the colored cells should be such that they alternate (one cell movement from red followed by one cell movement from green). Both the green and red cells stop when they've each traversed half the number of cells in the board (Notice that if the number of cells in the board is odd, the ending location of the green and red cells will be the same).

R	↓						↑
↓	↓						↑
↓	...						↑
↓						...	↑
↓						↑	↑
↓						↑	↑
↓						↑	G

Programming exercise 2

Write a program that repeatedly prompts the user to enter integer values (positive or negative) until the user enters a zero. The program will then display the following statistics based on the user input: the sum of the positive values entered, the sum of the negative values, the count of positive values, the count of negative values, and the overall average value. The zero entered to terminate the input should not be counted. Hint: the loop needs to have an if-else statement to check if the value entered is positive or negative, and accumulate the appropriate sum and the count.

Programming exercise 3

Write a program that simulates the flipping of a coin to determine how many flips are needed to obtain a given number of heads or tails in a row. Your program first prompts the user to enter the desired length of the sequence of flips with the same face of the coin – call that variable `sequenceLength`. For instance, if `sequenceLength` is 5, the program will repeatedly flip a coin until either 5 heads or 5 tails are encountered in a row. Flipping a coin can be simulated by generating a random integer of either 0 or 1 (0 can correspond to a head and 1 to a tail, or vice versa). Your program should output the generated sequence of flips (a series of H's and T's) until the desired number of heads or tails has been encountered.

Here are two sample runs of the program:

Sample run 1:

```
Enter the number of heads or tails I should look for in a sequence: 5
Flipping coins: HHTHHTHTHTHTHTTTHTHTHTHTHHHHH
It took 28 flips to encounter 5 heads in a row.
```

