

ISE 208

Midterm Exam 1

Practice Version – SOLUTIONS

NAME (please print legibly): _____

Your University ID Number: _____

- Please leave at least one seat between yourself and the person next to you.
- Please use a pen for all answers.
- No books or notes can be used during the exam, except for your “cheat sheet”.
- Any **CHEATING** will result in an F as well as being written-up on academic dishonesty. Don't place anything in a location where it might indicate that you are copying.

NO BS bonus: If you do not know the answer to a problem and leave it blank you will receive 1 point for each sub-part (e.g., a,b,c, etc.) that you leave blank. If you write anything in the space and it is wrong, you will receive a zero. Thus, the score for a completely blank exam is 5. Not all questions are of equal difficulty/points value, so read through the entire exam before beginning!

You will not lose any points for small coding details such as misspelling a method name, leaving out some default method arguments, or getting the order of method arguments wrong.

QUESTION	VALUE	SCORE
1	20	
2	15	
3	20	
4	10	
5	15	
6	20	
TOTAL	100	

1. (20 points)

Study the nested loop below. Rewrite it so that the outer loop is changed from a `for` loop to a `do...while` loop and the inner `for` loop is changed to a `while` loop. Your new nested loop should produce the same results as the original. You may declare any additional variables that you may need.

```
int sum = 0;

for (int i = 0; i <= 10; i++)
{
    for (int j = 0; j <= 10; j++)
    {
        sum += i;
    }
}
```

SOLUTION:

```
int sum = 0, i = 0; // i could also be initialized to 1

do
{
    int j = 0; // j can also be initialized to 1

    while (j <= 10)
    {
        sum = sum + i;
        j = j + 1;
    }

    i = i + 1;
} while (i <= 10);
```

2. (15 points)

Fill in the body of the method below, which computes and returns the value of the n th term of a geometric sequence defined by the terms

$$a, ar, ar^2, ar^3, \dots, ar^{n-1}$$

That is, the first term ($n = 1$) is a , the second term is ar , and so on.

```
public int sequence ( int a, int r, int n )
{
    int result = 0; // Just in case n is not a positive integer

    if (n > 0)
    {
        result = a;

        while (n > 1)
        {
            result = result * r;

            n = n - 1;
        }
    }

    return result;
}
```

3. (20 points)

The local Department of Motor Vehicles has asked you to write a program that grades the written portion of the driver's license exam. The exam has 20 multiple-choice questions. Here are the correct answers:

1. B	6. A	11. B	16. C
2. D	7. B	12. C	17. C
3. A	8. A	13. D	18. B
4. A	9. C	14. A	19. D
5. C	10. D	15. D	20. A

A student must correctly answer 15 of the 20 questions to pass the exam.

Write a class named `DriverExam` that holds the correct answers to the exam in an array. The class should also have an array field that holds the student's answers. The class should have the following methods:

- A constructor that takes the student's answers as its input
- `public boolean passed()` – Returns `true` if the student passed the exam, or `false` if the student failed.
- `public int totalCorrect()` – Returns the total number of correctly answered questions.
- `public int [] questionsMissed()` – Returns an `int` array containing the question numbers of the questions that the student missed (you may return an `ArrayList` instead, if you wish).

You may add any additional methods or instance variables that you wish. You may assume that the input (the student's answers) only contains the letters A, B, C, or D. You **do not** need to write a `main()` method or any other sort of driver, nor should you prompt the user for any input!

```

class DriverExam
{
    private char [ ] key;
    private char [ ] answers;

    public DriverExam ( char [ ] ans )
    {
        key = { 'B', 'D', 'A', 'A', 'C', 'A', 'B', 'A', 'C', 'D',
                'B', 'C', 'D', 'A', 'D', 'C', 'C', 'B', 'D', 'A' };

        answers = ans;
    }

    public boolean passed ()
    {
        return (totalCorrect() > 14);
    }

    public int totalCorrect ()
    {
        int correct = 0;

        for (int i = 0; i < key.length; i++)
        {
            if (key[i] == answers[i])
                correct++;
        }

        return correct;
    }
}

```

```
public int [ ] questionsMissed ()
{
    int size = key.length - totalCorrect(); // CORRECTED FROM PRIOR VERSION
    int [] missed = null;

    if (size < 1)
        return missed;
    else
        missed = new int [size];

    int pos = 0;

    for (int i = 0; i < key.length; i++)
    {
        if (key[i] != answers[i])
        {
            missed[pos] = (i + 1); // shift question 0 to 1, etc.
            pos = pos + 1;
        }
    }

    return missed;
}

} // end of DriverExam class
```

4. (10 points)

Match the test type with its description (2 pts each):

- (a) Unit test
- (b) Functional test
- (c) Parallel test
- (d) Stress test
- (e) Monkey test

1. Tests to see if a block of code works as it should
2. Simulates the worst possible workload
3. Tests system response to nonsensical input
4. Compares the functionality of a new system compared to the old system
5. Tests to see that a particular feature has been implemented

(a) 1

(b) 5

(c) 4

(d) 2

(e) 3

5. (15 points)

Define a method named `getLargestMultiple()`. This method should take two `int` arguments and return an `int` value. This method calculates the following value: the largest multiple of its second argument that is smaller than the first argument. For example, `getLargestMultiple(63, 5)` would return 60; 60 is the largest multiple of 5 that is less than 63 (the next multiple, 65, is too big). (**Hint:** Use either basic arithmetic or a loop.)

```
int getLargestMultiple (int first, int second)
{
    int result = 1;

    while (result < first)
    {
        result += second;
    }

    return result - second;
}
```

OR

```
int getLargestMultiple (int first, int second)
{
    int quotient = first / second;
    return quotient * second;
}
```

6. (20 points)

THE FOLLOWING PROBLEM IS EXTRA CREDIT

The method below takes an odd, positive integer n as its argument and prints a solid $n \times n$ square to the screen. For example, if n is 3, the method would print

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

Rewrite part or all of this method so that it prints a hollow square instead. That is, the interior rows should only contain stars for their leftmost and rightmost elements:

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

You may assume that n is always odd and positive.

```
public static void printSquare (int n)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            System.out.print("*");
        }

        System.out.println();
    }
}
```

```

public static void hollowSquare (int n)
{
    // Top row
    for (int top = 0; top < n; top++)
    {
        System.out.print("*");
    }
    System.out.println();

    // Middle rows
    for (int rows = 0; rows < (n - 2); rows++)
    {
        System.out.print("*");

        for (int mid = 0; mid < (n - 2); mid++)
        {
            System.out.print(" ");
        }

        System.out.println("*");
    }

    // Bottom row
    for (int bottom = 0; bottom < n; bottom++)
    {
        System.out.print("*");
    }

    System.out.println();
}

```