

ISE 208 Midterm 1

SBCS 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.

NO BS bonus: If you do not know the answer to a problem and leave it blank you will receive 1 point for each 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 0. The score for a completely blank exam is 14. You will not lose any points for small coding details or occasional misspellings.

QUESTION	VALUE	SCORE
1	20	
2	12	
3	18	
4	5	
5	15	
6	10	
7	20	
TOTAL	100	

1. (20 points)

Define a `PlayingCard` class that represents a single playing card. Every `PlayingCard` has a suit (represented by a `char`, either 'c', 'd', 'h', or 's') and a face value (an integer in the range 1-13, where 1 is an Ace and 11-13 represent the Jack, Queen, and King respectively).

Define the following methods for your `PlayingCard` class:

- A constructor that takes data for the face value and suit and assigns them to the appropriate instance variables.
- A method named `bValue()`. This method returns the value that the card would hold in a game of Blackjack. In Blackjack, aces have a value of 1, and all face cards have a value of 10. This method takes no arguments, and returns an `int` value.
- A `description()` method. This method takes no arguments, and returns a `String` containing a textual version of the card, e.g., "5 of Spades". Be sure to use words when describing an ace or a face card!

```

class PlayingCard
{
    private char suit;
    private int value;

    public PlayingCard (int faceValue, char suit)
    {
        value = faceValue;
        this.suit = suit;
    }

    public int bValue ()
    {
        if (value <= 9) return value;
        else return 10;
    }

    public String description ()
    {
        String result = null;

        if (value == 1) result = "Ace of ";
        else if (value == 11) result = "Jack of ";
        else if (value == 12) result = "Queen of ";
        else if (value == 13) result = "King of ";
        else result = "" + value + " of ";

        if (suit == 'C') result = result + "Clubs";
        else if (suit == 'D') result = result + "Diamonds";
        else if (suit == 'H') result = result + "Hearts";
        else result = result + "Spades";

        return result;
    }
}

```

2. (12 points)

Consider the following code fragment using `boolean` objects `a`, `b`, `c`, and `d`:

```
if (!c || d)
{
    if (a && c)
    { System.out.println(1); }
    else if (b)
    { System.out.println(2); }
    else
    { System.out.println(3); }
}
else if (!a == b)
{ System.out.println(4); }
else if (a)
{ System.out.println(5); }
else
{ System.out.println(6); }
```

- (a) Give values for `a`, `b`, `c`, and `d` that will cause the code fragment to display 1 to the screen. (4 pts)

`a = TRUE, b = any value, c = TRUE, d = TRUE`

- (b) Give values for `a`, `b`, `c`, and `d` that will cause the code fragment to display 3 to the screen. (4 pts)

`b = FALSE`, with `(a, c, d)` being one of `(T, F, T)`, `(T, F, F)`, `(F, T, T)`, `(F, F, T)`, or `(F, F, F)`

- (c) Give values for `a`, `b`, `c`, and `d` that will cause the code fragment to display 6 to the screen. (4 pts)

`a = FALSE, b = FALSE, c = TRUE, d = FALSE`

3. (18 points)

```
double mean;
```

For the variable declaration listed above, what is the value of `mean` after each of the following assignment statements? Write **ERROR** if a syntax error results.

```
mean = 100.0/40.0; // mean will be 2.5
mean = 100.0/40;  // mean will be 2.5
mean = 100/40;    // mean will be 2.0
```

```
int total;
```

For the variable declaration listed above, what is the value of `total` after each of the following assignment statements? Write **ERROR** if a syntax error results.

```
total = 100/40;           // total will be 2
total = 100.0/40.0;      // ERROR: must cast double to int
total = (int)(100.0/40.0); // total will be 2
```

4. (5 points)

Which of the following is **NOT** part of a method signature?

- (a) The name of the method
- (b) The method's return type
- (c) The types of the method's arguments
- (d) None of the above (these are all part of a method signature)

ANSWER: _____B

5. (15 points)

Hailstones are numbers generated according to the following sequence:

Let n_0 be a given positive integer. For all $i = 0, 1, 2, \dots$:

- if n_i is even, then $n_{i+1} = \frac{n_i}{2}$
- if n_i is odd, then $n_{i+1} = 3 * n_i + 1$
- if n_i is 1, the sequence ends.

For example, given the starting integer 77, the following sequence of hailstones will be generated:

77, 232, 116, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1

Fill in the body of the method below, which uses a loop to compute and print the hailstone sequence that results from the supplied input parameter. When this method is invoked, it should print out *all* of the hailstone values in the sequence (including the final 1).

```
void hailstone (int input)
{
    while (input > 1)
    {
        System.out.print(input + ", ");

        if (input % 2 == 0)
        {
            input = input / 2;
        }
        else
        {
            input = 3 * input + 1;
        }
    }

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

6. (10 points)

The array below is to be sorted using insertion sort. Fill in the contents of the array after each step in the insertion sort algorithm.

START: [412, 194, 66, 38, 7, 290, 994, 81, 361, 199]

[194, 412] [66, 38, 7, 290, 994, 81, 361, 199]

[66, 194, 412] [38, 7, 290, 994, 81, 361, 199]

[38, 66, 194, 412] [7, 290, 994, 81, 361, 199]

[7, 38, 66, 194, 412] [290, 994, 81, 361, 199]

[7, 38, 66, 194, 290, 412] [994, 81, 361, 199]

[7, 38, 66, 194, 290, 412, 994] [81, 361, 199]

[7, 38, 66, 81, 194, 290, 412, 994] [361, 199]

[7, 38, 66, 81, 194, 290, 361, 412, 994] [199]

[7, 38, 66, 81, 194, 199, 290, 361, 412, 994]

FINISH: [7, 38, 66, 81, 194, 199, 290, 361, 412, 994]

7. (20 points)

The method below examines two integer arrays that represent lottery numbers (one contains the winning numbers, and one contains a given player's selected numbers). This method examines the two arrays and returns the number of values that appear in both arrays. You may assume that the arrays are of equal length, and that each array contains only one copy of a given value (so, for example, an array cannot contain two copies of the same number).

For example, consider the arrays {7, 4, 9, 1, 3} and {4, 2, 9, 7, 3}. There are four values (3, 4, 7, and 9) that appear in both arrays, so the method would return 4 for these arguments.

```
public int countMatches (int [] array1, int [] array2)
{
    int matches = 0;

    for (int i = 0; i < array1.length; i++)
    {
        for (int k = 0; k < array2.length; k++)
        {
            if (array1[i] == array2[k])
            {
                matches++;
            }
        }
    }

    return matches;
}
```