First Name (Print): _____ **Last Name (Print):** _____

Student Number: _____

The Irving K. Barber School of Arts and Sciences
COSC 111 Midterm 1 Winter Term II 2016

Instructor: Dr. Bowen Hui

Monday, March 14, 2016

Time: 5:15pm - 6:30pm

Instructions:

This is a closed book exam. No calculators are allowed. Turn off your cell phone. Have your student ID card out on your desk.

This exam has 13 pages (including this cover page).

Question	Score	Possible Marks
Part 1		8
Part 2 Question 1		10
Part 2 Question 2		5
Part 2 Question 3		5
Part 3		12
Total		40

Part 1: Multiple Choice (8 points)

Circle one answer for each question below.

1. Suppose `int i = 5`, which of the following can NOT be used as an index for array `double[] t = new double[100]`?

   (a) `i`
   (b) `( int )( Math.random() * 100 )`
   (c) `10 + i`
   (d) `i + 6.5`
   (e) `0`

2. When you invoke a method with a parameter, the value of the argument is passed to the parameter. This is referred to as _____.

   (a) method invocation
   (b) pass by value
   (c) pass by reference
   (d) pass by name

3. Which of the following is correct?

   (a) `int[] a = new int[2];`
   (b) `int[] a = int[2];`
   (c) `int[] a = new int(2);`
   (d) `int a = new int[2];`
   (e) `int a() = new int[2];`

4. Assuming the loop body does not modify `i`, how many iterations will the following loop execute?

```
for (int i = 1; i <= n; i++)
{
  // iteration
}
```

   (a) 2*n
   (b) n
   (c) n - 1
   (d) n + 1

5. What is the output of the following fragment?

```
for (int i = 0; i < 15; i++)
  if (i % 4 == 1)
    System.out.print(i + " ");
```

(a) `1 3 5 7 9 11 13 15`

(b) `1 5 9 13`

(c) `1 5 9 13 16`

(d) `1 3 5 7 9 11 13`

(e) `1 4 8 12`

6. In the following code, you should fill in the blank with which return type?

```
public class Test
{
  public static void main(String[] args)
  {
    System.out.print("The grade is ");
    printGrade(78.5);

    System.out.print("The grade is ");
    printGrade(59.5);
  }

  public static __________ printGrade(double score)
  {
    if (score >= 90.0)
      System.out.println('A');
    else if (score >= 80.0)
      System.out.println('B');
    else if (score >= 70.0)
      System.out.println('C');
    else if (score >= 60.0)
      System.out.println('D');
    else
      System.out.println('F');
  }
}
```

(a) `int`

(b) `double`

(c) `boolean`

(d) `char`

(e) `void`

7. What is output of the following code:

```
public class Test
{
  public static void main(String[] args)
  {
    int list[] = {1, 2, 3, 4, 5, 6};

    for (int i = 1; i < list.length; i++)
      list[i] = list[i - 1];

    for (int i = 0; i < list.length; i++)
      System.out.print(list[i] + " ");
  }
}
```

(a) `1 2 3 4 5 6`

(b) `2 3 4 5 6 6`

(c) `2 3 4 5 6 1`

(d) `1 1 1 1 1 1`

8. What is the output of the following code?

```
double[] myList = {1, 5, 5, 5, 5, 1};
double max = myList[0];
int indexOfMax = 0;
for (int i = 1; i < myList.length; i++)
{
  if (myList[i] > max)
  {
    max = myList[i];
    indexOfMax = i;
  }
}
System.out.println(indexOfMax);
```

(a) `0`

(b) `1`

(c) `2`

(d) `3`

(e) `4`

Part 2: Short Answers (20 points)

Question 1. (10 points)

There are 3 parts to this question regarding the following code stored inside Tree.java:

```
 1  import java.util.Scanner;
 2  public class Tree
 3  {
 4    public static void main( String args[] )
 5    {
 6      Scanner input = new Scanner( System.in );
 7      System.out.println( "How tall do you want the tree to be?" );
 8      int height = input.nextInt();
 9
10      int row = 0;
11      for( int i=height; i>0; i-- )
12      {
13        row++;
14
15        // print leading spaces on a given line
16        for( int j=0; j<i; j++ )
17          System.out.print( " " );
18
19        // print stars up to middle column
20        for( int j=0; j<row; j++ )
21          System.out.print( "*" );
22
23        // print stars after middle column
24        for( int j=0; j<(row-1); j++ )
25          System.out.print( "*" );
26
27        System.out.println();
28      }
29    }
30  }
```

Part a. Suppose you run this program and you entered 4 as user input at the prompt. What is the output of this program?

Grading scheme:

- **[1 pt]** General pattern that is printed
- **[1 pt]** Exact number of spaces printed
- **[1 pt]** Exact number of stars printed

Part b. Consider how lines 16-17 in Tree.java on the previous page prints spaces. Replace these lines with a method called `prSpaces` that takes a given number as input and prints that number of spaces, so that this new method will work with the following revised version of Tree.java below. For this question, define `prSpaces` and show how you will call it inside the `main` method below by filling in the blank.

```
 1  import java.util.Scanner;
 2  public class Tree                                                    // revised
 3  {
 4    public static void main( String args[] )
 5    {
 6      Scanner input = new Scanner( System.in );
 7      System.out.println( "How tall do you want the tree to be?" );
 8      int height = input.nextInt();
 9
10      int row = 0;
11      for( int i=height; i>0; i-- )
12      {
13        row++;
14
15        // print leading spaces on a given line
16        prSpaces( ____________ );
17
18
19        // print stars up to middle column
20        for( int j=0; j<row; j++ )
21          System.out.print( "*" );
22
23        // print stars after middle column
24        for( int j=0; j<(row-1); j++ )
25          System.out.print( "*" );
26
27        System.out.println();
28      }
29    }
30  }
```

Grading scheme:

- **[1 pt]** Correct method header
- **[1 pt]** Correct method body
- **[1 pt]** Correct method invocation

Part c. Consider how lines 20-21 and lines 24-25 in Tree.java on the previous page prints stars. Replace these lines with a method called `prStars` that takes a given number as input and prints that number of stars, so that this new method will work with the following revised version of Tree.java below. For this question, define `prStars` and show how you will call it inside the `main` method below by filling in the blanks.

```
 1  import java.util.Scanner;
 2  public class Tree                                                    // revised
 3  {
 4    public static void main( String args[] )
 5    {
 6      Scanner input = new Scanner( System.in );
 7      System.out.println( "How tall do you want the tree to be?" );
 8      int height = input.nextInt();
 9
10      int row = 0;
11      for( int i=height; i>0; i-- )
12      {
13        row++;
14
15        // print leading spaces on a given line
16        prSpaces( ____________ );
17
18
19        // print stars up to middle column
20        prStars( ____________ );
21
22
23        // print stars after middle column
24        prStars( ____________ );
25
26
27        System.out.println();
28      }
29    }
30  }
```

Grading scheme:

- **[1 pt]** Correct method header
- **[1 pt]** Correct method body
- **[2 pts]** Correct method invocations

Question 2. (5 points)

Write a complete Java program that takes an integer number from the user and calculates the sum of the digits. Sample output:

```
Enter a number
12345
sum of digits = 15
```

The resulting sum was calculated based on 1 + 2 + 3 + 4 + 5.

Sample output:

```
Enter a number
789
sum of digits = 24
```

The resulting sum was calculated based on 7 + 8 + 9.

Grading scheme:

- **[1 pt]** Taking user input
- **[1 pt]** Extracting one digit at a time
- **[1 pt]** Updating the remaining digits
- **[1 pt]** Calculating the summation
- **[1 pt]** Specifying the stopping criteria

Question 3. [5 points]

Write the Java code inside the `main` method of a program with the following specifications. Define an array of strings called `verbs` with 5 elements in it. These elements are: `"play"`, `"run"`, `"dig"`, `"walk"`, and `"jump"`. Thereafter, use a loop to go through the elements of the array and display the following output:

```
See the children PLAY!
See the children RUN!
See the children DIG!
See the children WALK!
See the children JUMP!
```

Grading scheme:

- **[1 pt]** Declaring and creating the array
- **[1 pt]** Defining array elements
- **[1 pt]** Looping through the array
- **[1 pt]** Converting the strings to uppercase
- **[1 pt]** Displaying proper output

Part 3 Long Answer [12 points]

Write a Java program called `RecordGrades` that lets the user enter the name and the GPA score of a student. Part of the program has been completed for you below:

```
 1  import java.util.Scanner;
 2  public class RecordGrades
 3  {
 4    public static void main( String[] args )
 5    {
 6      Scanner input = new Scanner( System.in );
 7      System.out.println( "How many students are in the class?" );
 8      int numStudents = input.nextInt();
 9      .  .  .
10    }
11    .  .  .
12    // define additional methods
13  }
```

In the program above, there are two missing parts as shown by the "..." on lines 9 and 11. To complete the program, you will need to do the following:

- Inside `main`, create an array called `names` to store the students' first names. Make sure you indicate how many elements are needed when you create the array.
- Inside `main`, create an array called `scores` to store the students' GPA scores. Make sure you indicate how many elements are needed when you create the array.
- Define a method called `enterData` that lets the user enter all the names and scores available and stores them into the appropriate arrays. For convenience, you will want to use the same index for one student, so that when you enter the information for student i, `names[i]` stores this student's name and `scores[i]` stores this student's GPA.
- Define a method called `prRecords` that goes through all the elements in both arrays and displays them. With reference to the sample output shown below, you will need to display all the information with each row showing the name and score of a single student.
- Inside `main`, call `enterData` with the appropriate arguments.
- Inside `main`, call `prRecords` with the appropriate arguments.

A sample run of the successful completion of such a program would look like this, where the user enters the number of students in the class, enters the data for each student, and then see the entered information displayed:

```
How many students are in the class?
3
Enter the student's name: John
Enter the student's mark: 79
Enter the student's name: Jodie
Enter the student's mark: 99.5
Enter the student's name: Jeremy
Enter the student's mark: 54.5
John:   79.0
Jodie:  99.5
Jeremy: 54.5
```

To get full marks, you need to write a `complete` Java program that runs without errors if we were to type it in exactly as you have written in. Include any libraries that the program needs to import in order for it to work.

Grading scheme:

- **[1 pt]** Declaring and creating the `names` array
- **[1 pt]** Declaring and creating the `scores` array
- **[1 pt]** Invoking the `enterData` method correctly
- **[1 pt]** Invoking the `prRecords` method correctly
- **[4 pts]** Correctly written `enterData`: [1 pt] for header, [1 pt] for loop, [1 pt] for reading inputs, [1 pt] for storing inputs
- **[3 pts]** Correctly written `prRecords` [1 pt] for header, [1 pt] for loop, [1 pt] for displaying output
- **[1 pt]** Overall syntax

Common Methods and Definitions

- From the `Math` class:
  - `double random()`
    Returns a `double` value greater than or equal to 0.0 and less than 1.0
  - `int round( float a )`
    Returns the closest `int` value to `a`, with ties rounding up
  - `double pow( double a, double b )`
    Returns the value of `a` raised to the power of `b`
  - `double exp( double a )`
    Returns Euler's number e raised to the power of `a`
  - `double sqrt( double a )`
    Returns the correctly rounded positive square root of `a`
- From the `Character` class:
  - `boolean isDigit( char ch )`
    Returns true if `ch` is a digit, and false otherwise
  - `boolean isLetter( char ch )`
    Returns true if `ch` is a letter, and false otherwise
  - `boolean isLetterOrDigit( char ch )`
    Returns true if `ch` is a letter or a digit, and false otherwise
  - `boolean isLowerCase( char ch )`
    Returns true if `ch` is a lowercase letter, and false otherwise
  - `boolean isUpperCase( char ch )`
    Returns true if `ch` is an uppercase letter, and false otherwise
- From the `String` class:
  - `char charAt( int index )`
    Returns the character at position `index` of the string
  - `int indexOf( int ch )`
    Returns the index position within the string of the first occurrence of `ch`
  - `int lastIndexOf( int ch )`
    Returns the index position within the string of the last occurrence of `ch`
  - `int length()`
    Returns the number of characters in the string
  - `boolean startsWith( String prefix )`
    Returns true if the string starts with `prefix`, and false otherwise
  - `boolean endsWith( String suffix )`
    Returns true if the string ends with `suffix`, and false otherwise
  - `int compareTo( String str )`
    Returns an integer result for comparing two strings lexicographically
  - `int compareToIgnoreCase( String str )`
    Returns an integer result for comparing two strings lexicographically while ignoring case differences
  - `boolean equals( Object stringObject )`
    Returns true if the string is the same as `stringObject`, and false otherwise
  - `boolean equalsIgnoreCase( String str )`
    Returns true if the string is the same as `str`, and false otherwise
  - `String substring( int beginIndex )`
    Returns a new string that is a substring starting at position `beginIndex` of this string
  - `String substring( int beginIndex, endIndex )`
    Returns a new string that is a substring starting at position `beginIndex` and up to `endIndex` of this string
  - `String toLowerCase()`
    Converts all the characters in this string to lower case

  - `String toUpperCase()`
    Converts all the characters in this string to upper case

- From the `Scanner` class:
  - `int nextInt()`
    Scans the next token of the input as an `int`
  - `double nextDouble()`
    Scans the next token of the input as a `double`
  - `String next()`
    Returns the next complete token as a `String`
  - `String nextLine()`
    Returns everything in the current line as a `String`