Part I. Multiple Choice Questions (2 points each):

1. Which of the following statements is TRUE regarding a Java loop?

   (a) An overflow error can only occur in a loop.
   (b) A loop may have multiple exit points. ******
   (c) If a variable of type int overflows during the execution of a loop, it will cause an exception.
   (d) A continue statement doesn't transfer control to the test statement of the for loop.

2. What is the output when the following method has been executed?

```
void funcA()
{
   int i, s = 0;
   for (int k = 0; k < 5; k++) {
      i = 0;
      do {
         i++;
         s++;
      } while (i < k);
   }
   System.out.println(s);
}
```

   (a) 15
   (b) 14
   (c) 11 ******
   (d) 10

3. Which of the following statements is FALSE?

   (a) NumberFormatException is a runtime exception.
   (b) A checked exception is an exception that is checked at run time. ******
   (c) Every exception thrower must be either an exception catcher or an exception propagator.
   (d) An exception catcher must have a matching catch block for the thrown exception.

4. Which of the following loop bodies DOES compute the product from 1 to 10 (i.e., 1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 * 9 * 10)?

```
int s = 1;

for (int i = 1; i <= 10; i++)
{
   <What to put here?>
}

assert s == 1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 * 9 * 10;
```

(a) `s += i * i;`

(b) `s++;`

(c) `s *= i; ******`

(d) `s = s + s * i;`

5. What is stored in the variable `result` after the following code fragment has been executed?

```
int index = 0;
int result = 1;

while ( true )
{
   ++index;
   if ( index % 2 == 0 )
      continue;
   else if ( index % 5 == 0 )
      break;
   result *= 3;
}
```

(a) 81

(b) This loop does not terminate

(c) 9 ******

(d) 27

6. Which of the following is NOT part of a method signature?

(a) The method name

(b) The return data type ******

(c) The number of parameters

(d) The parameter types

Refer to the following classes for questions 7, 8, and 9.

```
public class A {
   private int x = 0;
   private int y = 0;

   public A(int a) {
      this(a, 5);
   }
   public A(int a, int b) {
      x = a;
      y = b;
   }
   public A(A a) {
      this(a.getX(), a.getY());
   }
   public int getX() {
      return x;
   }
   public int getY() {
      return y;
   }
   public void setX(int a) {
      x = a;
   }
   public void setY(int a) {
      y = a;
   }
   public static void swap(A a1, A a2) {
      A atemp = a1;
      a1 = a2;
      a2 = atemp;
   }
   public static void swap(AContainer aa1, AContainer aa2) {
      A atemp = aa1.a;
      aa1.a = aa2.a;
      aa2.a = atemp;
   }
}

public class AContainer {
   public A a;
   public AContainer(A a) {
      this.a = a;
   }
}
```

7. Which of the following statements is correct after executing the code below?

```
A a1 = new A(0);
A a2 = new A(a1);
a1.setX(10);
a1.setY(15);
```

   (a) `a1.getX()` returns 10, `a1.getY()` returns 15, `a2.getX()` returns 0, `a2.getY()` returns 5 ******
   (b) `a1.getX()` returns 10, `a1.getY()` returns 0, `a2.getX()` returns 5, `a2.getY()` returns 15
   (c) `a1.getX()` returns 0, `a1.getY()` returns 5, `a2.getX()` returns 10, `a2.getY()` returns 15
   (d) `a1.getX()` returns 5, `a1.getY()` returns 10, `a2.getX()` returns 0, `a2.getY()` returns 15

8. Which of the following statements is correct after executing the code below?

```
A a1 = new A(0);
A a2 = new A(a1);
a1.setX(10);
a1.setY(15);
A.swap(a1, a2);
```

   (a) `a1.getX()` returns 10, `a1.getY()` returns 15, `a2.getX()` returns 0, `a2.getY()` returns 5 ******
   (b) `a1.getX()` returns 10, `a1.getY()` returns 0, `a2.getX()` returns 5, `a2.getY()` returns 15
   (c) `a1.getX()` returns 5, `a1.getY()` returns 10, `a2.getX()` returns 0, `a2.getY()` returns 15
   (d) `a1.getX()` returns 0, `a1.getY()` returns 5, `a2.getX()` returns 10, `a2.getY()` returns 15

9. Which of the following statements is correct after executing this code?

```
A a1 = new A(0);
A a2 = new A(a1);
a1.setX(10);
a1.setY(15);
AContainer aa1 = new AContainer(a1);
AContainer aa2 = new AContainer(a2);
A.swap(aa1, aa2);
```

   (a) `aa1.a.getX()` returns 10, `aa1.a.getY()` returns 15, `aa2.a.getX()` returns 0, `aa2.a.getY()` returns 5
   (b) `aa1.a.getX()` returns 0, `aa1.a.getY()` returns 5, `aa2.a.getX()` returns 10, `aa2.a.getY()` returns 15 ******
   (c) `aa1.a.getX()` returns 5, `aa1.a.getY()` returns 10, `aa2.a.getX()` returns 0, `aa2.a.getY()` returns 15
   (d) `aa1.a.getX()` returns 10, `aa1.a.getY()` returns 0, `aa2.a.getX()` returns 5, `aa2.a.getY()` returns 15

10. What is the meaning of the second line of this code?

```
Formatter formatter = new Formatter(System.out);
formatter.format("%15.3f", 18.1453);
```

    (a) Format the floating point number 18.1453 using a field of width 18 and three decimal places.
    (b) Format the floating point number 18.1453 using a field of width 15 and three decimal places. ******
    (c) Format the floating point number 18.1453 putting 15 spaces in front of the number.
    (d) Format the floating point number 18.1453 reserving 15 characters for the number 18, and three decimal places.

11. Which of the following statements are TRUE regarding static class methods?

    I. Static class methods cannot access non-static class variables.
    II. Static class methods can be called by using either an object of that class type or the class name.
    III. Static class methods can call non-static private class methods directly.

    (a) I and III
    (b) I and II ******
    (c) II and III
    (d) I only

12. Which of the following is TRUE?

    (a) The multiple catch blocks should be listed in the order from general exception classes to more specialized ones.
    (b) If there is no exception, the finally block will not be executed.
    (c) If there are multiple catch blocks, only the first one matching the exception will be executed. ******
    (d) If there are multiple catch blocks, all blocks that match the exception will be executed.

13. What is the output of the code below if `s` is a `null` reference?

```
try {
   int i = s.indexOf( "null" );
   return i;
} catch (NullPointerException e) {
   System.out.print("Null pointer exception!");
} catch (Exception e) {
   System.out.print(" Exception!");
} finally {
   System.out.print(" Done!");
}
```

    (a) Null pointer exception! Exception!
    (b) Null pointer exception!
    (c) Null pointer exception! Done! ******
    (d) Null pointer exception! Exception! Done!

14. Which of the following is NOT TRUE about assertions?

    (a) Assertions are used to check for logical error in a program.
    (b) `assert <boolean expression>` throws `AssertionError` if `<boolean expression>` is evaluated to false.
    (c) When used correctly, assertion can catch programming flaws that cannot be caught by the Java compiler.
    (d) An assertion can be used to replace exception. ******

15. Which of the following is NOT TRUE about programmer-defined exceptions?

    (a) They are meant to replace assertions. ******
    (b) They inherit from the super class `Exception`.
    (c) They allow programmers to attach useful information to the exception objects.
    (d) They can be caught in a try-catch statement.

16. Which of the following statements is NOT TRUE for layouts and nested panels?

    (a) It is possible not to use any layout manager.
    (b) FlowLayout places GUI objects in the top-to-bottom, left-to-right order.
    (c) Layout managers and nested panels are particularly helpful when the window is resizeable.
    (d) All regions must be occupied if you use the BorderLayout manager. ******

17. What is produced by the following piece of code?

```
public class MyWindow {
   public static void main(String args[])
   {
      JFrame myWindow = new JFrame();
      myWindow.setSize(300, 200);
      myWindow.setTitle("CS 180");
      JLabel myLabel;
      JButton myButton = new JButton("Exit");
      myWindow.setVisible(true);
   }
}
```

    (a) A window with a label and a button.
    (b) Only a window. ******
    (c) Nothing.
    (d) A window with a button.

18. An action listener can be registered with:

    (a) a JMenuItem ******
    (b) a JMenuBar
    (c) a JMenu
    (d) All of the above

19. Examine the following code for a button (labeled Exit) handler.

```
class ButtonHandler implements ActionListener
{
   public void actionPerformed(JButton button)
   {
      System.out.println(button.getText());
   }
}
```

What is printed when an "Exit" button is clicked?

(a) We cannot create this handler since there is no constructor.

(b) `"Exit"` is printed.

(c) This code does not compile. ******

(d) Nothing is printed. The program will exit if the button is clicked.

20. The MouseListener interface specifies the following methods: mouseClicked, mouseEntered, mouseExited, mousePressed, and mouseReleased. In order to use a MouseListener in your code, you must provide an implementation for:

(a) None of the MouseListener methods

(b) All MouseListener methods ******

(c) At least the mouseClicked method

(d) Any one MouseListener method

This page left blank intentionally.

Part II. Programming Questions (60 points total):

1. (20 points) You have been given the following method which will take two operands of the type double, one operator (one of `+`, `-`, `*`, `/`), and evaluate the result. This method throws two exceptions which have been defined elsewhere.

```
public static double evaluate(double op1, double op2, char op) throws
   DivideByZeroException, IllegalOperatorException
```

Write a main method that will prompt for operand 1, operand 2, and an operator, and call `evaluate()` to evaluate the result. You should print out the result. Remember to handle all possible exceptions.

Note: Write your code on the next page.

```
public static void main(String args[])
{
   // complete the main() method as specified
}
```

Solution for programming question 1:

```
public static void main(String args[])
{
    double op1, op2, result;
    char op;
    String opString;

    try
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter operand 1: ");
        op1 = sc.nextDouble();

        System.out.println("Enter operand 2: ");
        op2 = sc.nextDouble();

        System.out.println("Enter operator (+-*/): ");
        opString = sc.next();
        op = opString.charAt(0);

        result = evaluate(op1, op2, op);
        System.out.println("The result is: " + result);
    }
    catch(InputMismatchException e)
    {
        System.out.println("Input Mismatch Exception");
    }
    catch(IllegalOperatorException e)
    {
        System.out.println("Illegal Operator Exception");
    }
    catch(DivideByZeroException e)
    {
        System.out.println("Divide By Zero Exception");
    }
}
```

2. (20 points) A prime number (or a prime) is a natural number which has exactly two distinct natural number divisors: 1 and itself. An infinitude of prime numbers exists, as demonstrated by Euclid around 300 BC. The first 8 prime numbers are: 2, 3, 5, 7, 11, 13, 17, 19.

   According to the definition, we can find a very simple method to verify whether a number `x` is a prime number: we just enumerate all the numbers from 2 to `x-1`, and if `x` is divisible by any of them, `x` is not a prime number; otherwise, `x` is a prime.

   Your should write a class, call it `PrimeNumber` to prompt the user for a number `n` and output the first `n` prime numbers. This class should include the `main()` method and another method `printPrime()`. `printPrime()` takes an integer number `n` as an argument, and print the first `n` prime numbers. The `main()` method should keep accepting user input for `n` and calling `printPrime(n)` for each `n` until the user enters a 0.

   Note: Write your code on the next page. It must be a complete class.

```
import java.util.*;

public class PrimeNumber
{
    // complete the PrimeNumber class as specified
}
```

Solution for programming question 2:

```
public class Prime
{
    public static void printPrime(int n)
    {
        boolean prime;
        int t = 0;
        int i = 2;

        while (t < n)
        {
            prime = true;
            for (int j = 2; j < i; j++)
                if (i % j == 0)
                {
                    prime = false;
                    break;
                }
            if (prime)
            {
                t++;
                System.out.println(i);
            }
            i++;
        }
    }

    public static void main(String[] args)
    {
        int n;
        Scanner reader = new Scanner(System.in);

        System.out.println("Enter an integer (0 to terminate): ");
        n = reader.nextInt();
        while (n != 0)
        {
            printPrime(n);
            System.out.println("Enter an integer (0 to terminate): ");
            n = reader.nextInt();
        }
    }
}
```

3. (20 points) You have been given parts of a class called `BinaryDisplay`. This class has the following data members (a partial list):

```
private JButton oneButton;      // button labeled ONE
private JButton zeroButton;     // button labeled ZERO
private JButton clearButton;    // button labeled CLEAR
private JTextField outputLine;  // text field for displaying the binary string
private String  binaryString;   // string of 0's and 1's being constructed
```

Assume that the constructor is properly written to display a window with the 3 buttons properly aligned and the text field under the buttons. Also assume that the buttons have been properly associated with an action listener. Write the `actionPerformed()` method which will:

(a) append a 0 to the string in the text field if the ZERO button is pressed

(b) append a 1 to the string in the text field if the ONE button is pressed

(c) show a message, "STOP! You cannot enter more than 32 bits", if either the ZERO or the ONE button is clicked when there are already 32 digits

(d) clear the text field if the CLEAR button is pressed

Note: Write your code on the next page.

```
public void actionPerformed(ActionEvent e)
{
    // complete the method as specified
}
```

Solution for programming question 3:

```
public void actionPerformed(ActionEvent e)
{
    JButton clickedButton = (JButton) e.getSource();

    String buttonText = clickedButton.getText();

    if (binaryString.length() < 32 || buttonText.equals("clear"))
    {
        if(buttonText.equals("1"))
        {
                binaryString += "1";
        }
        else if(buttonText.equals("0"))
        {
                binaryString += "0";
        }
        else if(buttonText.equals("clear"))
        {
                binaryString = "";
        }

        outputLine.setText(binaryString);
    }
    else{
        JOptionPane.showConfirmDialog(this,
           "STOP! You cannot enter more than 32 bits.");
    }
}
```