TOTAL MARKS: 100 marks

INSTRUCTIONS:

Answer FIVE questions from Questions 1 – 6. These code-writing questions are worth 12 marks each and should be answered in the answer booklet supplied.

Answer Question 7, this is worth 40 marks, and consists of 20 multiple choice parts. Each of these should be answered in pencil on the computer-readable multiple choice answer sheet supplied.

The exam is CLOSED BOOK - books and notes are not permitted.

PLEASE NOTE

Examination candidates may only bring authorised materials into the examination room. If a supervisor finds, during the examination, that you have unauthorised material, in whatever form, in the vicinity of your desk or on your person, whether in the examination room or the toilets or en route to/from the toilets, the matter will be reported to the head of school and disciplinary action will normally be taken against you. This action may result in your being deprived of any credit for this examination or even, in some cases, for the whole unit. This will apply regardless of whether the material has been used at the time it is found.

Therefore, any candidate who has brought any unauthorised material whatsoever into the examination room should declare it to the supervisor immediately. Candidates who are uncertain whether any material is authorised should ask the supervisor for clarification.
Instructions

Make sure that you first read the entire paper and plan how to approach the paper. In general, you can get full marks on a code-writing question more easily than getting six multiple-choice questions correct, so you may wish to consider attempting the code-writing questions first.

In addition, the questions, including the multiple-choice questions are not in any particular order of difficulty, so answer the ones that you find straightforward first.

Don’t forget to include your name and student number on the multiple-choice answer sheet, and double-check your student number as it will be read by computer.

Do not spend too much time on the multiple choice.
1.

The Dominican flag is a red, white and blue flag with the following design (where the numbers show the relative sizes of the different coloured components).

```
<table>
<thead>
<tr>
<th>10</th>
<th>3</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blue</td>
<td>Red</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>White</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Red</td>
<td>Blue</td>
<td></td>
</tr>
</tbody>
</table>
```

Write a method

```java
public void drawFlag()
```

that will cause the flag to be drawn on the screen with a height of 150 pixels.

Your method should create and use a `SimpleCanvas` (as used in lectures and laboratories) to draw on. The method should also create the necessary `Color` objects. The method should not rely on any instance variables.

(12 marks)
2.

Suppose that we need to develop a simple class called `Country` for use in a geographical database system. The class will need variables to hold (a) the name of a country, which contains mainly alphabetic characters, (b) its capital city as alphabetic characters, (c) area in square km as a double, and (d) population as an integer. The only constructor for this class requires four arguments, the name, capital, population, and area. The methods needed are `getName()` which returns the name of the country, `getCapital()` which returns the name of the capital city, `getArea()` which returns the area in square km, `getPopulation()` which returns the population size, and `setPopulation(popSize)` which changes the size of population to the value `popSize`.

(a) Write out a skeleton of the `Country` class, clearly showing all variable declarations and method definitions. Their respective types, signatures, and arguments must be included. You are NOT required to include any code in the methods.

(2 marks)

(b) In an existing `Geography` class, the following method is required:

```java
public static String largestCountry(Country[] countryList)
```

It should return the name of the country that has the largest land area. Write out the method in detail.

(5 marks)

(c) Suppose that we are also interested in finding the country that is most densely populated (i.e., having the most number of people per square km). Write a method

```java
public static String mostDenselyPopulatedCountry(Country[] countryList)
```

that returns the name of the most densely populated country.

(5 marks)
3.

(a) Write a method

```java
public int indexOfMax(int[] a)
```

that will return the index of the largest value in the one-dimensional array `a`. (6 marks)

(b) The value \( \ln 2 \) (that is, the natural log of 2) can be approximated by an infinite series as follows:

\[
\ln 2 = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \cdots
\]

Write a method

```java
public double approximateLn2(int n)
```

that approximates \( \ln 2 \) using the first \( n \) terms of this series. (6 marks)

4.

Consider a class `MyDictionary` that can represent a subset of the English language. The argument of the single constructor of this class is an array `String[] words` that represent the words to store in the dictionary. Note that the words are case-sensitive and not ordered in the array.

The constructor requires a helper method from the `Sort` class to sort the array in normal alphabetical order.

Write a method for the `Sort` class using the `InsertionSort` algorithm, with the following signature

```java
public static void alphaInsertionSort(String[] words)
```

(12 marks)
5.

Imagine a pizzeria in which the pizza chef is careless when tossing olives onto a pizza base. In fact he throws them completely randomly in the square area containing the pizza.

One can actually get a simple approximation to π by counting the proportion of the olives that hit the pizza.

A pizza of diameter 1 has an area of $\pi/4$, while the area of the entire square containing the pizza is 1. Therefore the proportion of olives hitting the pizza multiplied by 4 is an approximation to $\pi$. In the example, 20 out of 28 olives have hit the pizza and so this gives the (rather poor) approximate value of $4 * 20 / 28 = 2.857$

Write a method, 

```java
public static double pi(int olives)
```

that simulates the random throwing of olives on to a pizza base in order to approximate the value of $\pi$. Your method should use an object from the library class `java.util.Random` to generate a sequence of random olive positions inside a suitable square, and keep track of how many land on the pizza. The client will specify the number of olives required, and you should return the corresponding approximation. Your method may **not** refer to `Math.PI`.

You may wish to use the following method from the class `java.util.Random`:

```java
public double nextDouble()
```

Returns the next pseudorandom, uniformly distributed double value between 0.0 and 1.0 from this random number generator's sequence.
6.

The Game of Life is a 2-dimensional cellular automaton. However there are also 1-dimensional cellular automata, where the world is simply a 1-dimensional strip of cells, each of which is either occupied or vacant. A 15 cell world is shown below with marks signifying occupied cells.

Each cell has just two neighbours which are the cells one step to the right or left of it. We assume that the strip “wraps around” so that the left-hand edge is joined to the right-hand edge (forming a circular strip).

```
  y y y y
```

At each time-step the world evolves according to the following rules: any cell with 0 or 2 occupied neighbours remains in its current state, while any cell with a single occupied neighbour changes state (that is, from occupied to unoccupied or vice versa).

For example, for the configuration shown above, each cell has the following number of occupied neighbours.

```
  1 0 1 1 0 2 0 1 0 1 0 1 0 0 0
```

and so applying the rule to all cells simultaneously the world evolves to its next state:

```
  ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●
```

Write a method

```java
public boolean[] nextGeneration(boolean[] map)
```

that takes a boolean array representing the current state of the cellular automaton and computes and returns a boolean array representing the next generation.

(12 marks)
7. Multiple Choice Questions [2 marks each]

(1) What is the value of mystery(89) if mystery is defined as follows:

```java
public static String mystery(int x) {
    if (x==0)
        return "";
    else
        return mystery(x/2) + Integer.toString(x%2);
}
```

A. "0000000"
B. (*) "1011001"
C. "1111000"
D. "1001101"
E. "1000001"

(2) Suppose that we are attempting to sort the following array into increasing order:

\{10, 19, 2, 17, 3, 11, 7\}

If we use QuickSort, then what does the array contain after the first “partitioning” stage (assume that the fence is chosen to be the first element of the array).

A. \{2, 3, 7, 10, 19, 17, 11\}
B. \{10, 2, 3, 7, 19, 17, 11\}
C. \{19, 17, 11, 10, 2, 3, 7\}
D. (*) \{3, 7, 2, 10, 17, 11, 19\}
E. \{2, 3, 7, 10, 11, 19, 17\}

(3) What is the main reason that a programmer would choose to use checked exceptions such as `java.io.FileNotFoundException`?

A. To provide information to the compiler that permits it to generate faster running code than if it had to account for all possible error types.
B. To help maintain object integrity by preventing direct client manipulation of the internal stucture of an object.
C. (*) To force client code to explicitly provide error-handling code to deal with certain commonly occurring error situations.
D. To allow the programmer to formally express certain conditions that he/she expects to hold at certain stages during the runtime of a program.
E. To cause the program to halt if a client accidentally (or deliberately) uses a method with invalid parameters.
(4) What output will the method call `show(5)` produce on the terminal window if the method `show` is defined as follows:

```java
public void show(int n) {
    int num = 0;
    for (int i=1; i<n; i++) {
        for (int j=0; j<i; j++) {
            System.out.print(num++);
            System.out.print(" ");
        }
        System.out.println(" ");
    }
}
```

A. 1
   2 3
   4 5 6
   7 8 9 10

B. 0
   0 0
   0 0 0
   0 0 0 0

C. 0
   0 1
   0 1 2
   0 1 2 3

D. (*)
   0
   1 2
   3 4 5
   6 7 8 9

E. 0
   1 2
   3 4 5
   6 7 8 9
(5) If a, b and c are variables of boolean type, then which of the following expressions is true whenever one or two of a, b and c are true, and false otherwise.

A. \((a \& b) || (a \& c) || (b \& c)) \& \& !((a \& b) \& \& c)\)
B. \(a || b) \& \& (a || c) \& \& (b || c)\)
C. \((a \& b \& \& \!c) || (a \& \!b \& \& c) || (!a \& \& b \& \& c)\)
D. (*) \((a \neq b) || (a \neq c) || (b \neq c)\)
E. \((a || b || c) || !((a \& b) \& \& c)\)

(6) Consider the following method for calculating the Lucas numbers

```java
public int lucas(int n) {
    if (n == 1) return 1;
    if (n == 2) return 3;
    return lucas(n-1)+lucas(n-2);
}
```

What does the method call `lucas(5)` return?

A. 5
B. 7
C. 9
D. (*) 11
E. 13

(7) If we are searching for an item in a sorted array, containing 64 items, then approximately how many probes must we make in the worst case if we use binary search. (As described in lectures, a “probe” refers to examining an item of the array to see if it equal to the one we are looking for.)

A. Exactly 1
B. (*) About 7
C. About 32
D. About 64
E. About 4096 \((64 \times 64)\)
(8) Consider the following sequence of statements where *BankAccount* is as given in the lectures.

```java
BankAccount b1 = new BankAccount("Bill Gates", 12345, 0);
BankAccount b2 = new BankAccount("Bill Gates", 12345, 0);
BankAccount b3 = new BankAccount("Larry Ellison", 54321, 0);
b1.deposit(200);
b3 = b2;
b2.deposit(350);
b3.withdraw(150);
```

What are the values of

\[ \text{b1 == b2} \]

and

\[ \text{b1.getBalance()} == \text{b3.getBalance()} \]

respectively?

A. true and true
B. false and false
C. true and false
D. (*) false and true
E. false and 200.

(9) In the Java statement

```java
double y = Math.pow(x, 2);
```

what is *pow()*?

A. A method belonging to the object *Math*
B. (*) A class method belonging to the class *Math*
C. A class variable from the class *Math*
D. An instance variable of the class *Math*
E. A constant value defined in the class *Math*
(10) Consider the following method \( m \).

```java
public boolean m(String s)
{
    int i = 0;
    int sl = s.length();
    int hl = sl / 2;

    while (s.charAt(i) == s.charAt(sl-(i+1)) && i < hl)
        i++;

    return i == hl;
}
```

What does this method calculate?

A. (*) determines that \( s \) is a palindrome by returning true.
B. it will always generate an IndexOutOfBoundsException.
C. it isn’t correct Java syntax and it will not compile.
D. returns the number of letters that don’t match in a string.
E. None of the above.

A palindrome is a string that reads the same backwards and forwards.

(11) Consider the following method \( \text{int mystery(int a, int b)} \), where you may assume that \( a \geq 0, b > 0 \).

```java
public int mystery(int a, int b) {
    if (b == 1) {
        return a;
    } else {
        return a + mystery(a, b-1);
    }
}
```

What does this method calculate?

A. It is a recursive way to calculate \( a - b \).
B. (*) It is a recursive way to calculate \( a \times b \).
C. It is a recursive way to calculate \( a \% b \).
D. It is a recursive way to calculate \( a^b \).
E. It is a recursive way to calculate \( a + b \).
(12) Suppose that an array int[] f contains frequency data for some application. The cumulative frequency array is obtained by replacing each element in the array by the sum of itself and all of the previous elements in the array.

In other words, the cumulative frequency array is

\{f[0], f[0]+f[1], f[0]+f[1]+f[2], ...\}

Which of the following three methods will correctly transform the frequency array f into a cumulative frequency array?

// Method 1
public void cumulate(int[] f) {
    for (int i=1; i<f.length; i++) {
        f[i] = f[i] + f[i-1];
    }
}

// Method 2
public void cumulate(int[] f) {
    for (int i=0; i<f.length; i++) {
        for (int j=0; j<i; j++) {
            f[i] = f[i] + f[j];
        }
    }
}

// Method 3
public void cumulate(int[] f) {
    for (int i=f.length-1; i>0; i--) {
        for (int j=0; j<i; j++) {
            f[i] = f[i] + f[j];
        }
    }
}

A. All of them.
B. None of them.
C. 2 and 3 only.
D. (*) 1 and 3 only.
E. 1 and 2 only.
(13) Which one of the following statements about a method with a `void` return type is true?

A. A `void` method cannot have any `return` statements.
B. A `void` method must have exactly one `return` statement.
C. A `void` method must have at least one `return` statement.
D. A `void` method must have at most one `return` statement.
E. (*) A `void` method can have zero, one or more `return` statements.

(14) Suppose that a class `ClassA` has a method with the following signature:

```java
public static void method()
```

and we further suppose that variable `a` is declared and created in another class, `ClassB`, as follows:

```java
ClassA a = new ClassA();
```

Consider the following three Java statements that occur in a method belonging to `ClassB`. Which one is a valid Java statement?

1. `ClassA.method();`
2. `a.method();`
3. `method();`

A. 1 only.
B. 2 only.
C. (*) only 1 and 2.
D. only 1 and 3.
E. all of them.
(15) In a program dealing with family trees, objects of the class \texttt{Person} are used to represent people. Part of the code is as follows:

```java
public class Person {

    // This person’s father and mother.
    private Person father;
    private Person mother;

    // This person’s children if any, null otherwise.
    private Person[] children;

    // irrelevant code omitted

    public int mystery() {
        if (this.children == null) {
            return 1;
        } else {
            int total = 0;
            for (int i=0; i<children.length; i++) {
                total = total + children[i].mystery();
            }
            return total;
        }
    }
}
```

If \texttt{Person bill} represents a particular individual Bill, then what does \texttt{bill.mystery()} calculate?

A. How many children Bill has.
B. How many descendants Bill has (i.e. his children, grand-children, great-grand-children etc).
C. How many people there are in Bill’s extended family (i.e. Bill, his children, grand-children, great-grand-children etc).
D. (*) How many members of Bill’s extended family have no children of their own.
E. None of the above.
The remaining questions deal with the following code, which defines two classes `Point` which represents points with integer co-ordinates, and `Rectangle` which represents rectangles whose corners are `Points`. You may assume that `Math.min(a, b)` and `Math.max(a, b)` find the minimum and maximum respectively of two integer values `a` and `b`.

```java
public class Point {
    private int x;
    private int y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public Point() {
        this(0, 0);
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }
}

public class Rectangle {
    // topLeft and botRight are the locations of the top
    // left and bottom right corners of the Rectangle
    // in conventional Java graphics co-ordinates

    private Point topLeft;
    private Point botRight;

    public Rectangle(Point tL, Point bR) {
        if (tL.getX() >= bR.getX() || tL.getY() >= bR.getY()) {
            throw new IllegalArgumentException("Illegal Rectangle");
        }
        topLeft = tL;
        botRight = bR;
    }

    public Rectangle(int x, int y, int width, int height) {
        if (width <= 0 || height <= 0) {
            throw new IllegalArgumentException("Illegal Rectangle");
        }
    }
}
```


topleft = new Point(x,y);
bottoright = new Point(x+width,y+height);

public Rectangle(int width, int height) {
    if (width <= 0 || height <= 0) {
        throw new IllegalArgumentException("Illegal Rectangle");
    }
    toppoint = new Point();
bottoright = new Point(width,height);
}

public int area() {
    int result = (bottoright.getX()-toppoint.getX());
    return result * (bottoright.getY()-toppoint.getY());
}

public boolean query(Point p) {
    boolean b = (p.getX() >= toppoint.getX());
b = b && (p.getX() <= bottoright.getX());
b = b && (p.getY() >= toppoint.getY());
return (b && (p.getY() <= bottoright.getY()));
}

public Rectangle biggest(Rectangle other) {
    if (area() > other.area())
        return this;
    else
        return other;
}

public static Rectangle intersection (Rectangle r1, Rectangle r2) {
    int tlx = Math.max(r1.topLeft.getX(), r2.topLeft.getX());
tly = Math.max(r1.topLeft.getY(), r2.topLeft.getY());
int brx = Math.min(r1.bottoright.getX(), r2.bottoright.getX());
bry = Math.min(r1.bottoright.getY(), r2.bottoright.getY());
return new Rectangle(new Point(tlx,tly), new Point(brx,bry));
}
(16) Which of the following are legal ways to construct a Rectangle.

1. `new Rectangle(10, 10, 25, 100);`
2. `new Rectangle(Point(10, 10), Point(25,100));`
3. `new Rectangle(15, 90);`

A. 1 and 2 only
B. (*) 1 and 3 only
C. 2 and 3 only
D. All of them.
E. None of them.

(17) Suppose that a program starts as follows:

```
Point p1 = new Point(100,100);
Point p2 = new Point(200,200);
Rectangle r1 = new Rectangle(p1,p2);
Rectangle r2 = new Rectangle(100,100,100,100);
Rectangle r3 = r1.biggest(r2);
```

How many objects of the class `Point` have been constructed in total?

A. 2
B. 3
C. (*) 4
D. 5
E. 6

(18) What does the method `r.query(p)` return?

A. It returns `true` if the point `p` is strictly inside (i.e. inside but not on the edge) of the rectangle `r`, and `false` otherwise.
B. It returns `true` if the point `p` is to the right of the rectangle `r`, and `false` otherwise.
C. It returns `true` if the point `p` is outside, or on the edge, of the rectangle `r` and `false` otherwise.
D. (*) It returns `true` if the point `p` is inside, or on the edge, of the rectangle `r` and `false` otherwise.
E. It returns `true` if the point `p` is to the left of the rectangle `r` and `false` otherwise.
(19) What does \( x \) contain after the following sequence of statements is compiled and run?

Point \( p1 = \text{new Point}(50,100); \)
Point \( p2 = \text{new Point}(120,200); \)
Rectangle \( r1 = \text{new Rectangle}(p1,p2); \)
Rectangle \( r2 = \text{new Rectangle}(10,10,80,90); \)
int \( x = r1.\text{biggest}(r2).\text{area}(); \)

A. 5000
B. 5600
C. 7000
D. (*) 7200
E. 24000

(20) What does the following line of code do?

\( \text{Rectangle[]} r = \text{new Rectangle}[10]; \)

A. It creates a variable \( r \) with the value \text{null}.
B. It creates a variable \( r \) referring to an array containing 10 references each of which refers to a newly created \text{Rectangle}.
C. It creates a \text{Rectangle} with corners \((0,0)\) and \((10,10)\)
D. It creates a variable \( r \) referring to an array containing 10 references each equal to 0.
E. (*) It creates a variable \( r \) referring to an array containing 10 references each equal to \text{null}.