Please fill in your Student Number and Name.

Student Number : ________________________________

Name:

____

____

Student Number:

____

University of Cape Town ~ Department of Computer Science

Computer Science 1015F ~ 2007

Final Examination

Question	Max	Mark	Interna	External
1	15			
2	15			
3	15			
4	15			
5	4			
6	12			
7	12			
8	12			
TOTAL	100			

Marks **: 100**

Time **: 3 hours**

Instructions:

a) Answer all questions.

b) Write your answers in the space provided.

c) Show all calculations where applicable.

Question 1 [15]

Read the following program carefully.

```
import java.util.Scanner;

public class Question1
{
   public static final int MAXL = 4;
   public static void main(String[] args)
   {
      Scanner keyboard = new Scanner(System.in);
      System.out.println("Enter a word:");
      String a = keyboard.next();
      a = (a.length() <= MAXL) ? a : a.substring(0,MAXL);

      for(int i=0; i<a.length(); i++)
         System.out.print(a);
   }
}
```

a) Give an example of a named constant from the program above. [1]

b) What is an advantage of using the named constant in the program above? [1]

c) Give an example of a boolean expression used in the program above. [1]

d) Rewrite the **for** loop in the program above as a **while** loop. Just write down the replacement code, not the entire program. [4]

e) Describe what this program does. [2]

f) Write down the exact output of the program after the user enters the line “hi fred!” at the prompt. [2]

g) Write down the exact output of the program after the user enters “100” at the prompt. [2]

h) Rewrite the *conditional operator* in the program above as an **if** statement. Just write down the replacement code, not the entire program. [2]

Question 2 [15]

Examine the following program carefully.

```
import java.util.Scanner;
public class Question2
{
   public static void main(String[] args)
   {
      Scanner keyboard = new Scanner(System.in);
      System.out.println("Enter a value:");
      int value = keyboard.nextInt();
      int row=1;
      while(row!=value*2)
      {
         for(int stars=1; stars<value*2; stars++)
         {
            if ((row==value)&&(stars==value))
            {
               System.out.print(' ');
               continue;
            }
            System.out.print('*');
         }
         System.out.println();
         row++;
      }
      for(int spaces=0; spaces<value-1; spaces++)
         System.out.print(' ');
      System.out.println('*');
   }
}
```

a) What are *nested loops*? [1]

____

b) Identify a nested loop in the program above. [1]

____

c) Write down the *exact* output of the program after the user types in the value “2” at the prompt. [3]

d) This program can have an infinite loop if the user enters the wrong value at the prompt.

i. What is an *infinite loop?* [1]

ii. Which values would cause an infinite loop? [1]

iii. How would you correct the program so that infinite loops would not occur? [1]

e) Would a **do-while** loop be more useful than a **while** loop for this example? Explain your answer. [2]

f) What is the difference between a **continue** statement and a **break** statement in a loop? [2]

g) Assume that the **continue** statement in the program is replaced with a **break** statement. Give an example of the output that shows the effect of this change. [3]

Question 3 [15]

Consider the following program fragment:

```
public AList doAction ( int a, int b, int c )
{
   if (a < b) {
      if (b < c)
         return new AList (a, b, c);
      else if (a < c)
         return new AList (a, c, b);
      else
         return new AList (c, a, b);
   } else {
      if (a < c)
         return new AList (b, a, c);
      else if (b < c)
         return new AList (b, c, a);
      else
         return new AList (c, b, a);
   }
}
```

a) What does this code do? [2]

b) **AList** is the name of a class. What is a class? [1]

c) **doAction** is a method. What is a method? [2]

d) Suppose we wish to test this method using statement coverage.

i. How many sets of test input values are needed? [1]

ii. Provide one such set of values. [3]

e) How does path coverage differ from statement coverage? [2]

f) The code needs to be compiled before it can be executed.

i. What is a compiler? [1]

ii. What is a debugger? [1]

g) Provide 2 examples of high-level languages other than Java. [2]

Question 4 [15]

You need to calculate the roots of a quadratic polynomial. In the context of this problem, answer the following questions.

a) Describe an algorithm to calculate the roots. Check for real roots - do not calculate complex roots. [4]

b) Describe how you would model your solution using object-oriented programming (mention the major classes and members and the purpose of each). [3]

c) How would we deal with complex roots using OOP? [1]

d) Write a method **getRootOne** to return one of the roots, given instance variables for the coefficients. Remember that $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. [4]

e) What special syntax must be used in Java to activate encapsulation? [1]

f) If some of your variables are declared as **private**, you need an accessor to access them from outside the class. Write an accessor method for one of the coefficients. [2]

Question 5 [4]

You are given the following code fragment.

```
public class Value
{
   private String name;
   private int number;
   public void change ( int val )
   {
      val = this.number;
      System.out.println ("in change val = " + val);
   }
   public void What ( Data dd )
   {
      dd.number = this.number;
   }
   public void set ( String s, int i )
   {
      name = s;
      number = i;
   }
}

public class Exam
{
   public static void main ( String[] args )
   {
      Value result1 = new Value(),
            result2 = new Value();
      result1.set ("Exam1", 60);
      result2.set ("Exam2", 80);

      int val = 90;
      result1.change(val);
      System.out.println ("in Main val = " + val);

      result2.What(result1);
      System.out.println ("in Main result1.number = " +
                          result1.number);
      System.out.println ("in Main result2.number = " +
                          result2.number);
   }
}
```

a) What is the output? [4]

____

____

____

Question 6 [12]

a) Write Java code to define a 1-dimensional array called **val** that contains the following 7 values (6, 0, 1, -5, 2, -3, -6), and write a loop that processes those values as follows:

- All zeroes have 10 added to them,
- All negatives are changed to positives and
- All positives are multiplied by 3

  E.g., (6, 0, 1, -5, 2, -3, -6) becomes (18, 10, 3, 5, 6, 3, 6). [5]

Note: Do not put your code into a class or method – just write the Java statements.

b) Write code that prints these values in reverse order i.e., (18, 10, 3, 5, 6, 3, 6) becomes (6, 3, 6, 5, 3, 10, 18). [1]

Note: Do not put your code into a class or method – just write the Java statements.

c) Write Java code to set the elements of an array to the values indicated below. *Two* loops must be sensibly used and no values may be read in from the keyboard. Define all variables used. The code should work for any square array. [6]

Note: Do not put your code into a class or method – just write the Java statements.

-1	0	0	0	0
0	-1	1	1	1
0	2	-1	2	2
0	2	4	-1	3
0	2	4	6	-1

Question 7 [12]

Complete the **two** classes given below:

- A **Run** class that calls the function **BlockAverage** to find the average water consumption for the period specified.
- A **Water** class that has a predefined method called **fill** (as shown below) that assigns a value to each element of the array that holds the water consumption for each month, for 5 years.

You must write:

- A function called **PartialMonthSum** that, given a month, a start year and an end year, returns the total water consumption for the period concerned – for example, for month 2, start year 1 and end year 3, the sum returned is [(6.1+5.1+5.0) = 16.2].
- A function called **BlockAverage** that, given a start month and an end month and a start year and an end year, returns the average water consumption for this block. This function must call the function **PartialMonthSum** repeatedly – for example, for start month 2, end month 4, start year 1 and end year 3:

  ((6.1+5.1+5.0) + (6.5+5.5+5.6) + (6.9 +5.6+6.1)) /9

  = (16.2 + 17.6 + 18.6) / 9

  = 8.73

Wet	M 0	M 1	M 2	M 3	M 4	M 5	M 6	M 7	M 8	M 9	M 10	M 11
Year 0	5.2	5.4	5.6	6.0	6.2	6.5	7.0	8.1	7.5	7.1	6.9	5.5
Year 1	5.3	5.8	6.1	6.5	6.9	7.2	8.1	9.2	8.0	7.5	7.2	6.0
Year 2	4.9	5.0	5.1	5.5	5.6	5.8	6.2	6.5	6.8	7.1	7.0	6.5
Year 3	4.7	4.8	5.0	5.6	6.1	6.2	7.0	7.5	6.8	6.2	5.8	5.4
Year 4	4.5	4.6	4.8	5.1	5.3	5.4	5.6	5.0	5.5	4.9	4.7	4.6

```
public class Water
{
   private float [][] wet;

   public void fill ()
   {
     //Assume this code has been provided
   }

   public Water ()
   {
     //Assume that this constructor has been provided.
   }
```

a) `// PartialMonthlySum` [4]

b) `// BlockAverage` [6]

```
} // end of Water class
```

c) `// The Run class – fill in the missing line(s).` [2]

```
public class Run()
{
   public static void main (String [] args)
   {
      int average;
      int startyr = 1;
      int endyr = 3;
      int startmonth = 2;
      int endmonth = 4;
      Water w = new Water();
      w.fill();
```

```
      System.out.println("Av =", average);
   }
}
```

Question 8 [12]

Study the following incomplete class definitions. Use the concepts of *inheritance* and *polymorphism* in your answers to the questions that follow.

```
class Car
{
   private String name;
   private int engineSize;
   public Car (String n, int size) //parameterized constructor
   {
      name = n;
      engineSize = size;
   }
   public String toString()
   {
      return ("Car name"+ name + "Engine size" + engineSize);
   }
}

class StationWagon extends Car
{
   private float luggageCapacity;
   public StationWagon (…); //parameterized constructor
   public String toString(…);
}

class FourbyFour extends StationWagon
{
   private int numberGears;
   public FourbyFour (…); //parameterized constructor
   public String toString(…); //assume this is given
}
```

a) Write parameterized constructors for **StationWagon** and **FourbyFour**. (The constructor for the class **Car** has been given in the definition of **Car**). [5]

```
// StationWagon Constructor
// FourbyFour constructor
```

b) Write the **toString** method for **StationWagon**. (The **toString** method for the class **Car** has been given in the definition of **Car**). You may assume that the **toString** method for **FourbyFour** has been provided. [2]

c) Write a **main** method that creates at least one parameterized object of each class (**Car**, **StationWagon** and **FourbyFour**). Print out all the data of each of these objects from inside a loop (thus illustrating the general method of doing this). [5]

```
public static void main (String [] args)
{
```

}