CSI 1100 / 1500
Fall 2004
Introduction to Computer Science I
Final Examination

Duration : 3 hours 09:30 December 9, 2004

Professors: Alan Williams, Daniel Amyot Page 1 of 13

Full name (please print): _____

Student number: _____

Instructions – please read carefully!

- **Fill out the identification information above in ink.**
- This exam is closed-book. **No books, papers, calculators, or other electronic devices are permitted.**
- There are 7 questions in this exam, some of which have sub-parts. Answer the questions in the space provided. **Answers written in pencil will not be re-graded even if there is a marking error.**
- The marks allocated to each question are provided. Not all questions are worth the same, so plan your time accordingly. The exam has 100 marks in total, which represents 55% of your final grade.
- Algorithms in pseudocode should use the format from the course notes.
- You can use the back of pages for calculations and rough work, as well as page 12. Pages 12 and 13 can be detached and will not be graded.
- Les réponses en français sont acceptées.

Question	Marks	Marks received	Question	Marks	Marks received
1	12		5	15	
2	8		6	25	
3	15		7	10	
4	15		**Total**	100	

Question 1A) (4 marks)

In this question, use only the following (**pseudocode**) Boolean expressions:

- comparison operators: <, >, =, ≤, ≥, and ≠
- Boolean connectors: NOT, AND, OR
- arithmetic operators: +, -, *, /, and MOD (modulo)
- variable names and constants.

Use parentheses where necessary. Do **not** use Java syntax!

Environment Canada will report a humidex value as part of a weather forecast if the temperature (T) is greater than or equal to 30 degrees, if the temperature is greater than or equal to 25 degrees and the humidity (H) is greater than 35%, or the temperature is greater than or equal to 20 degrees and the humidity is greater than or equal to 65%.

Write a Boolean expression that is true if Environment Canada will report a humidex value, and false otherwise.

Answer:

Question 1B) (4 marks)

Consider the following Java program :

```
MyClass[] obj;
int index;

obj = new MyClass[2];
index = 15;
while( index > 2 )
{
   obj[index % 2] = new MyClass( );
   index = index / 2;
}
// Line X
```

i) How many instances of MyClass are created during the execution of this program? (2 marks)
Answer:

ii) How many instances of MyClass are still accessible at Line X? (2 marks)
Answer:

Question 1C) (4 marks)

Consider the following classes:

```
class Foo
{
  private int x1;
  public static int x2;
  public static Bar x3;

  public Foo(int x4)
  {
       ...
  }
}
```

```
class Bar
{
  public int x5;

  public static int x6()
  {
       ...
  }

  public Foo x7()
  {
       ...
  }
}
```

Suppose that the following instructions are used in the `main()` method in a class `Test`. Each choice should be considered independently – as it if were in its own `main()` method. Circle the letter of the statement which causes a compilation error.

(a)
```
Foo[] a = new Foo[5];
a[4] = new Foo(-1);
```

(b)
```
Foo f = Bar.x7();
```

(c)
```
Foo.x2 = Bar.x6();
```

(d)
```
int k = Foo.x3.x5;
```

(e)
```
Bar b = new Bar();
Foo f = b.x7();
```

Question 2: (8 marks)

Here is a program that uses recursion.

```
class Football
{
  public static void main(String[ ] args)
  {
    char[] t = {'G', 'e', 'e', '-', 'G', 'e', 'e'};
    Rec(t, t.length - 1);
  }

  public static void Rec(char[] var, int i)
  {
    if (i < 0)
    {
      ; // do nothing
    }
    else
    {
      if ( (var[i] > 'A') && (var[i] < 'Z') )
      {
        System.out.print( (char) (var[i] - 'A' + 'a') );
      }
      else
      {
        if ( (var[i] > 'a') && (var[i] < 'z') )
        {
          System.out.print( (char) (var[i] - 'a' + 'A') );
        }
        else
        {
          ; // do nothing
        }
      }
      Rec(var, i - 1);
    }
  }
}
```

What is printed during the execution of the method `main` in this class

Answer :

Question 3: (15 marks)

Translate the algorithm on page 13 to a Java method.

Question 4: (15 marks)

Olympic 10 metre platform diving events are scored as follows. Each judge watches an athlete's dive, and then submits a score for the dive (out of 10). The dive is also previously assigned a "degree of difficulty" (DD) based on the complexity of the particular dive (example: forward 2 1/2 somersault in the tuck position has DD = 2.7). The highest and lowest scores are discarded, and the remaining scores are added together and then multiplied by the degree of difficulty to determine the total dive score.

Write an algorithm that will compute an athlete's total dive score from an array of scores submitted by N judges, for a dive of degree of difficulty DD. Remember to include comments with your algorithm!

Question 5: (15 marks)

The lower right sub-matrix of a matrix is formed by selecting one element position (row and column) and excluding all elements that are to the left or above the selected element. For example, in the matrix M below, if we select $M_{11} = 5$, the matrix S is the lower right sub-matrix.

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad S = \begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix}$$

Write a Java method that will take a matrix of integers ***M***, and a row and column index, and returns a new matrix that is the lower right sub-matrix of ***M*** formed from that position. The header of the method is as follows:

```
public static int[][] subMatrix( int[][] m, int theRow, int theCol )
```

Question 6: (25 marks)

In this question, you will create a class `Experiment` that represents a record of some sort of scientific experiment. In order to verify that the results of an experiment are repeatable, there is a class `Trial` that contains the results from one run of an experiment. An experiment will then include a number of `Trial` objects.

The class `Trial` stores a result that was measured during an experiment, and the duration that the experiment took, measured in milliseconds. The class `Trial` has already been implemented. A UML class diagram for the class Trial is as follows:

Trial
- result : double - duration: int
+ Trial(theResult : double, theDuration : double) + getResult( ) : double + getDuration( ) : int

On the next two pages, you will fill in the methods for `Experiment`. Your `Experiment` class should provide four public methods and/or constructors that would permit the following class `TestExperiment` to execute:

```
class TestExperiment
{
   public static void main (String[] args)
   {
      Experiment anExperiment;

       anExperiment = new Experiment( 2 );
      anExperiment.addTrial( new Trial( 99.1, 10000 ) );
      anExperiment.addTrial( new Trial( 97.1, 11000 ) );
      anExperiment.addTrial( new Trial( 94.1, 12000 ) );
      Experiment.setPredictedResult( 98.6 );
      anExperiment.print();
   }
}
```

Executing `main()` would result in the following being printed on the screen :

```
No more trials can be added to the experiment.
Trial 0: Result 99.1, duration 10000 (within 0.5 of prediction)
Trial 1: Result 97.1, duration 11000 (within 1.5 of prediction)
```

Question 6: (continued)

Complete the Class `Experiment` on this page and the next.

```
class Experiment
{
    // FIELD DECLARATION(S): (4 marks)

    // CONSTRUCTOR: (5 marks)
    // Takes one integer parameter representing the maximum number of trials that can be put into
    // the experiment.

    // METHOD setPredictedResult: (4 marks)
    // Method parameters: a double that is the predicted result of the experiment.
    // Result:  none
```

Question 6: (continued)

```
    // MODIFIER METHOD addTrial: (6 marks)
    // Method parameters: a Trial object that should be added to the Experiment.
    // Results:  will print a message if the experiment has no room to store additional trials
    // (see page 8 for message format)
    // Modified:  the Experiment object
```

```
    // METHOD print: (6 marks)
    // Method parameters: (none)
    // Returns: (none)
    // This method prints the result and duration  of each trial, along with the absolute value
    //  of the difference from the predicted result.
    // See the TestExperiment sample output on Page 8 for exact format.
```

```
} // End of class Experiment
```

Question 7: (10 marks)

Suppose that you have an array of integers that is already known to be sorted in ascending order, and to contain no duplicate values. A binary search on such an array is done by locating the middle element of an array and comparing the search value with that element. If we have not found the value for which we are searching, we can choose the appropriate sub-interval (either the left or the right side) to restrict the search.

Write a recursive method that will perform a binary search on an array `a` for the value `findMe` between positions `startIndex` and `endIndex`. You can assume that both positions are less than the length of the array. The method should return `true` if the value is contained in the array, and `false` otherwise.

The header of the method should be the following:

```
public static boolean binarySearch(int[] a, int findMe,
                                   int startIndex, int endIndex)
```

This page is for calculations and other work, and can be detached. This page will not be marked.

Algorithm for Question 3 (this page can be detached)

GIVENS:

Base:	(a logarithm base, known to be > 0)
Operand:	(an array of integers for which to find the integer logarithm)
N	(the number of values in array Operand)

RESULT:

IntLog:	(an array of N integer logarithms for the values in array Operand; a value of -1 is returned if the logarithm does not exist)

INTERMEDIATES:

Index	(array index)
Value	(used for repeated divisions)
Count	(counts number of times an operand can be divided by base)

HEADER: IntLog ← Logarithms(Base, Operand, N)

BODY: