The German University in Cairo
Dpt. for Media Engineering and Technology
Prof. Dr. Slim Abdennadher, Assoc. Prof. Dr. Georg Jung

March 31, 2012

CSEN 202: Introduction to Computer Programming

Spring 2012

Midterm exam

Model Solutions

Instructions. Please read carefully before proceeding.

(a) The duration of this exam is **120 minutes**.

(b) Non-programmable calculators are allowed.

(c) No books or other aids are permitted for this test.

(d) This exam booklet contains a total of **10 pages**, including this one.

Exercise 1 Primitive types (11 Marks)

Types and expressions

Assume the following statements to be executed in sequence. Give the exact output of the following printing statements. If a statement is incorrect, please indicate the error.

(a) `short s = -128; System.out.println (s);` (1 Mark)

Solution:

```
-128
```

(b) `byte b = (byte) s; System.out.println (b);` (1 Mark)

Solution:

```
-128
```

(c) `b += 1; System.out.println (b);` (1 Mark)

Solution:

```
-127
```

(d) `s <<= 2; System.out.println (s);` (1 Mark)

Solution:

```
-512
```

(e) `b = (byte) s; System.out.println ((double) b);` (1 Mark)

Solution:

```
0.0
```

(f) `System.out.println (""+ s + b);` (1 Mark)

Solution:

```
-5120
```

(g) `int i = -1; System.out.println (i & s);` (1 Mark)

Solution:

```
-512
```

(h) `int j = 1; System.out.println (j || s);` (1 Mark)

Solution:

Error: The operator `||` is only defined on **`boolean`**.

(i) `byte c = -8; System.out.println ((c >> 1) + c);` (1 Mark)

Solution:

```
-12
```

(j) `boolean h = (0 = 1)?true:false; System.out.println (h);` (1 Mark)

Solution:

Error: A single equality sign ("=") is the assignment operator. Assignments require a variable on the left side.

(k) `float f = 42.0; System.out.println (f == 42);` (1 Mark)

Solution:

Error: The literal "`42.0`" is of type **`double`** and cannot be implicitly cast into a **`float`**.

Exercise 2 Refactoring (6 Marks)
Conditional

(a) Rewrite the following code segment to eliminate all unnecessary checks (3 Marks)

```
if (a <= b && a <= c)
  result = a;
else if (b <= a && b <= c)
  result = b;
else if (c <= a && c <= b)
  result = c;

System.out.println(result);
```

Solution:

```
if (a <= b && a <= c)
  result = a;
else if (b <= c)
  result = b;
else
  result = c;

System.out.println(result);
```

(b) Rewrite the previous code segment to use only the print statement with a nested conditional expression enclosed. (3 Marks)

Solution:

```
System.out.println((a <= b && a <= c) ? a : ((b <= c) ? b : c));
```

Exercise 3 Tracing (8 Marks)
Board

(a) Give the exact output of the following Java program-segment into the table below (use one cell for each character) (4 Marks)

```
for (int i = 0; i < 6; i++) {
  for (int j = 0; j < 6; j++)
    System.out.print ((i + j) % 2 == 0 ? "␣" : "*");
  System.out.println ();
}
```

Put your answer into this table (one character per cell):

	*		*		*			
*		*		*				
	*		*		*			
*		*		*				
	*		*		*			
*		*		*				

(b) Give the exact output of the following Java program-segment into the table below (use one cell for each character). Assume the argument `n` set to a value of 5. (4 Marks)

```
public static void sAndCr (int n) {
  for (int i = 0; i < n; i++, System.out.println ())
    for (int j = 0; j < n; j++)
      System.out.print ((i == j || i + 1 == n - j) ? "*" : "␣");
}
```

Put your answer into this table (one character per cell):

*				*				
	*		*					
		*						
	*		*					
*				*				

Exercise 4 Conditionals (13 Marks)
Geometry calculator

(a) Write a Java program that calculates the perimeter or the area of a geometric shape based on the choice of the user (the choices are *circle*, *rectangle*, and *square*). The user should enter a character or a number indicating what he needs to calculate and another one for the type of the shape shape. (13 Marks)

- Your program must have a complete infrastructure consisting of a class, a `main`-method, and any helper methods you might need.
- Take the input by using a `Scanner`.
- To interpret the input you must use a **`switch`**-statement.
- Ask for the right input data (*e. g.*, side, radius, . . .) in each specific case.

Solution:

```
import java.util.Scanner;

public class Geometry {

  public static void main (String[] args) {
    Scanner sc = new Scanner (System.in);
    System.out.println
      ("Please␣enter␣\'p\'␣for␣perimeter␣or␣\'a\'␣for␣area:␣");
    char c = sc.next ().charAt (0);
    System.out.println
      ("Please␣enter␣\'c\'␣for␣circle,␣\'r\'␣for␣rectangle␣or␣\'s\'␣for␣square:␣");
    char t = sc.next ().charAt (0);
    double area = 0;
    double perimeter = 0;
    switch (c) {
    case 'a':
      switch (t) {
      case 'c':
        System.out.println ("Enter␣radius:");
        int r = sc.nextInt();
        area = 3.14 * r * r;
        break;
      case 'r':
        System.out.println ("Enter␣side:");
        int side1 = sc.nextInt();
        int side2 = sc.nextInt();
        area = side1 * side2;
        break;
      case 's':
        System.out.println ("Enter␣side:");
        int side = sc.nextInt();
        area = side * side;
        break;
      }
      System.out.println ("Area␣=␣" + area);
      break;
    default:
      switch (t) {
      case 'c':
        System.out.println ("Enter␣radius:");
        int r = sc.nextInt ();
        area = 2 * 3.14 * r;
        break;
      case 'r':
        System.out.println ("Enter␣sides:");
```

```
            int side1 = sc.nextInt ();
            int side2 = sc.nextInt ();
            area = 2 * (side1 + side2);
            break;
          case 's':
            System.out.println ("Enter side:");
            int side = sc.nextInt ();
            area = 4 * side;
            break;
          }
          System.out.println ("Perimeter = " + perimeter);
        }
      }
    }
```

Exercise 5 Nested loops (12 Marks)
S-factorial

The s-factorial (super-factorial) of a positive whole number n (s-fact(n)) is defined as

$$\text{s-fact}(n) = 1! \cdot 2! \cdot \ldots \cdot n!$$

with

$$n! = 1 \cdot 2 \cdot \ldots \cdot n.$$

(a) Write a Java program to find the *s-factorial* of a positive integer n. (12 Marks)

- Your program must have a complete infrastructure consisting of a class, a `main`-method, and any helper methods you might need.
- In particular: Write at least a method `sfact` and method `main` to test it.
- Take the input by using a `Scanner`.

Solution:

```
import java.util.Scanner;

public class SFactorial {

  public static void main (String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println ("Enter␣a␣number:");
    long n = sc.nextLong ();

    /* Testing the iterative version (sufficient for full Marks) */
    System.out.println ("The␣s-factorial␣of␣" + n + "␣is␣" + sfact(n));

    /* Testing the recursive version (alternative) */
    System.out.println ("The␣s-factorial␣of␣" + n + "␣is␣" + sfactRec(n));
  }

  /* Iterative version (sufficient to receive full marks) */
  public static long sfact (long n) {
    long result = 1;
    for (int i = 1; i <= n; i++) {
      long factorial = 1;
      for (int j = 1; j <= i; j++)
        factorial *= j;
      result *= factorial;
    }
    return result;
  }

  /* Recursive version (alternative, coded defensively) */
  public static long sfactRec (long n) {
    return (n <= 1) ? factRec (1) : factRec (n) * sfactRec (n - 1);
  }

  public static long factRec (long n) {
    return (n <= 1) ? 1 : n * factRec (n - 1);
  }
}
```

Exercise 6 Iterations and conditionals (10 Marks)

LCM

(a) Write a method `lcm (int a, int b, int c)` that takes three positive integers a, b, and c and returns the least common multiple (lcm) of the three. (10 Marks)

Hint. Copy the values of `a`, `b`, and `c` into three auxiliary variables (*e. g.*, `x`, `y`, and `z`). Then repeatedly find the smallest of the three auxiliary variables and add its base value to it (*i. e.*, if `x` contains the smallest value, then add `a` to `x`, if `y` contains the smallest value, then add `b` to `y`, *etc.*). When all three auxiliary variables are equal, you have found the least common multiple.

Example. For example, take $a = 12$, $b = 15$, and $c = 20$. The algorithm can be illustrated by the following table:

Iteration	x	y	z
1st	12	15	20
2nd	24		
3rd		30	
4th			40
5th	36		
6th		45	
7th	48		
8th			60
9th		60	
10th	60		

Hint. Refer to Exercise 2 for a way of how to find the least out of three values.

Solution:

```
public static int lcm (int a, int b, int c) {
  int x = a;
  int y = b;
  int z = c;

  while (!(x == y && y == z)) {
    if (x <= y && x <= z)
      x += a;
    else if (y <= z)
      y += b;
    else
      z += c;
  }
  return x;
}
```

Bonus Exercise 7 Recursion (6 Marks)
GCD

The Euclidean algorithm to find the greatest common divisor of two integer numbers can be concisely written as

$$\gcd(n, m) = \begin{cases} m & \text{if } n = 0 \\ \gcd(m \mod n, n) & \text{otherwise} \end{cases}$$

where $n \mod m$ stands for the remainder of an integer division (*i. e.*, "$n \mod m$" $\equiv$ "`n % m`").

(a) Write a *recursive* method that calculates the greatest common divisor of two integer numbers **(4 Marks)**

(b) Two extra bonus-Marks can be earned if your algorithm uses the conditional operator (expression) instead of a conditional statement. **(2 Marks)**

Solution:

(Only the method `gcd` had to be implemented.)

```
public class Gcd {

  public static void main (String[] args) {
    System.out.println (gcd (220, 325));
  }

  public static int gcd (int n, int m) {
    return (n == 0) ? m : gcd (m % n, n);
  }
}
```