German University in Cairo
Faculty of Media Engineering and Technology
Prof. Dr. Slim Abdennadher

17. June 2007

Concepts of Programming Languages
Spring term 2007

Final Exam

Bar Code

Instructions: Read carefully before proceeding.

1) No books or other aids are permitted for this test.
2) This exam booklet contains 14 pages, including this one. Three extra sheets of scratch paper are attached and have to be kept attached. **Note that if one or more pages are missing, you will lose their points. Thus, you must check that your exam booklet is complete**.
3) Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem or on the three extra sheets and make an arrow indicating that. **Scratch sheets will not be graded unless an arrow on the problem page indicates that the solution extends to the scratch sheets**.
4) **Duration of the exam:** 3 hours
5) When you are told that time is up, stop working on the test.

Good Luck!

Don't write anything below `;-)`

Exercise	1	2	3	4	5	6	7	8	9	10	$\sum$
Possible Marks	8	6	8	10	8	6	8	4	6	6	70
Final Marks											

Exercise 1 (8 Marks)

a) Java includes three modifiers that can be used to attribute a degree of visibility to data members, methods, and classes: `private`, `public`, and `protected`.

1. Put these three modifiers in order of increasing visibility.
2. Describe how each modifier affects the visibility of data members, methods, and classes.
3. Briefly explain what effect using no modifier has on visibility

Solution:

1. increasing visibility: `private`, `protected`, `public`
2. - `private`: the data or methods can be accessed only by the declaring class;
   - `protected`: a protected data member or a protected method in a public class can be accessed by any class in the same package or its subclasses (even if the subclasses are in a different package);
   - `public`: the class, data, or method is visible to any class in any package.
3. No modifier: can be accessed directly from any class within the same package, but not from other packages.

b) How does overloading differ from parametric polymorphism in Haskell? Explain and give concrete examples.

Solution:

A parametric function can take any type as parameter. If we want to do the same using overloading, we have to write a function for each type. Overloading is applied when different operations on different types use the same name. Examples: parametric polymorphism: map function in Haskell. overloading: the equality operator == in Haskell, which must be instantiated in order to be usable. (type classes). It behaves differently for each type.

c) Explain why changing the order of rules in a Prolog program can change the program's behavior.

Solution:

When attempting to satisfy a goal, the Prolog interpreter starts at the beginning of the database and uses the first rule it finds that will unify with the goal. If there is more than one way to satisfy a goal, the interpreter will find the first one. Similarly, if rules have side effects, evaluating them in different orders will alter program behavior. With some orders (e.g., if the base case of a recursion is not encountered first), the interpreter can fall into an infinite regression.

Exercise 2 (6 Marks)

Define a Prolog predicate `take/3` that selects elements from a list by their position and collects them in a result list. For example, the goal `take([2,4,5],[a,b,c,d],L)` should produce the answer `L=[b,d]`. You can assume that the numbers in the first list are strictly increasing, that is, your implementation does not have to care about situations like `take([1,1,2],...)` or `take([2,4,3],...)`.

Solution:

```
take([],_,[]).

take(_,[],[]).

take([H|T],[H1|T1],[H1|T2]) :-
       H = 1, remove(T,T3),
       take(T3,T1,T2).

take([H|T],[H1|T1],T2) :-
       H \= 1, remove([H|T],T3),
       take(T3,T1,T2).


remove([],[]).

remove([H|T], [H1|T1]) :-
    H1 is H - 1,
    rem_one(T,T1).
```

Exercise 3 (8 Marks)

a) Write a Haskell function `remove n e l` that removes the first `n` occurrences of an element `e` from a list `l`.

For example:

```
remove 0 1 [1,2,3] --> [1,2,3]
remove 2 'a' ['a','b','c','a','d','a'] --> ['b','c','d','a']
```

b) Give the type of the function `remove`.

Solution:

```
-- Remove the first occurrence of an element from a list
remove1 :: Eq a => a -> [a] -> [a]
remove1 _ [] = []
remove1 e (x:xs)
 | e == x = xs
 | otherwise = x : remove1 e xs

-- Remove a number of occurrences of an element from a list using the function remove1
remove :: Eq a => Int -> a -> [a] -> [a]
remove 0 _ xs = xs
remove n e xs = remove (n - 1) e (remove1 e xs)

-- Second Version: Faster Method

removeF :: Eq a => Int -> a -> [a] -> [a]
removeF 0 _ xs = xs
removeF _ _ [] = []
removeF n e (x:xs)
 | e == x = removeF (n - 1) e xs
 | otherwise = x : removeF n e xs
```

Exercise 4 (10 Marks)

Given the following grammar to describe expressions

```
Expr  -> Int | Expr + Expr | Expr=Expr
```

The grammar can be represented in Haskell as follows:

```
data Expr = N Int
          | Plus Expr Expr
          | Equal Expr Expr
          | Not Expr
          deriving Show
```

For example,

- the expression 5 + 6 can be represented in Haskell as `Plus N 5 N 6`.
- The equality 5 + 1 = 3 + 3 can be represented as `Equal Plus N 5 N 1 Plus N 3 N 3`.

The aim of the exercise is to evaluate these expressions. Since an expression can be evaluated to either an integer or to a Boolean value, we will use the following datatype for the output of the function:

```
data Val = I Int
         | B Bool
         deriving Show
```

a) Implement a function `eval` that takes as parameter an expression and returns the value of the expression. **Hint:** Use the predefined function `not` to evaluate expressions of the form `Not e`.

   For example:

   ```
   eval Plus N 5 N 6 --> I 11
   eval Equal Plus N 5 N 1 Plus N 3 N 3 --> B true
   ```

b) Give the type of the function `eval`.

Solution:

```
eval :: Expr -> Val
eval (N i)        = I i
eval (Plus e e')  = I (i+j)   where (I i,I j) = (eval e,eval e')
eval (Equal e e') = B (i==j)  where (I i,I j) = (eval e,eval e')
eval (Not e)      = B (not b) where (B b) = eval e
```

Exercise 5 (8 Marks)

Implement a function `bag` that takes a list as arguments and returns a list of pairs, where each pair consists of an element of the list and its number of occurrences.

For example:

```
bag [1,3,4,1,3,2,1] --> [(1,3),(3,2),(4,1),(2,1)]
```

Hint: The use of the higher-order function `filter` will lead to a solution consisting of two patterns.

Solution:

```
bag :: Eq a => [a] -> [(a, Int)]
bag [] = []
bag (x:xs) = (x, 1 + length (filter (==x) xs)) : bag (filter (/=x) xs)
```

Exercise 6 (6 Marks)

Strings in C can be seen as an array of characters.

a) Implement a C function with the following prototype that copies a string

```
void stringcopy(char original[], char kopie[]);
```

DO NOT USE `strcpy`**!**

Solution:

```
% First Solution
void stringcopy(char original[],
                char kopie[])
{
    unsigned int i = 0;

    while (original[i] != '\0')
    {
        kopie[i] = original[i];
        i++;
    }
    kopie[i] = '\0';
}


% Second Solution
void stringcopy(char original[],
char kopie[])
{
    unsigned int i = 0;

    while ((kopie[i] = original[i]) != '\0')
    {
        i++;
    }
}

% Second Version!

void stringcopy(char original[],
                char kopie[])
{
    unsigned int i = 0;

    while (kopie[i] = original[i])
    {
        i++;
    }
}
```

b) Implement a second function that does the same job but now using pointers. The function has the following prototype

```
void stringcopy(char *original, char *kopie);
```

DO NOT USE `strcpy`**!**

Solution:

```
void stringcopy(const char *original,
char *kopie)
{
    while ((*kopie++=*original++) != '\0')
;
}
```

Exercise 7 (8 Marks)

Write the function

```
char *mirror(char *string);
```

that gets one string as its argument and returns a pointer to the beginning of a (new) memory location that contains the original string an its reverse. For example `mirror("hello")` returns the pointer to a memory location that holds the string `"helloolleh"`.

Hint: Use `malloc` function to allocate memory. Please use the standard library function `strlen()` for your implementation. It is declared in the standard header file `string.h` and its prototype is

```
int strlen(const char *string);
```

Solution:

```
char *mirror(const char *string)
{
     int laenge = strlen(string);
     char *tmp;
     char *neu;


     tmp = neu = (char *) malloc((2 * laenge + 1) * sizeof(char));

         if (neu == NULL)
             return NULL;


         while (*string != '\0')
               *tmp++ = *string++;

         for (--string; laenge > 0; --laenge)
               *tmp++ = *string-- ;

         *tmp = '\0';

         return neu;
}
```

Exercise 8 (4 Marks)

Consider the following Java classes `C1`, `C2`, and `C3`. Write down the output that would be generated by the `main()` method associated with class `C1`.

```
public class C1 extends C2 {

    public C1() {
         System.out.println("C1’s constructor is invoked");
    }

    public void hello() {
         System.out.println("Hello from C1");
    }

    public static void main(String[] args) {
         new C1().hello();
         new C2().hello();
    }
}


class C2 extends C3 {

    public C2() {
         System.out.println("C2’s constructor is invoked");
    }
}

class C3 {

    public C3() {
         System.out.println("C3’s constructor is invoked");
    }

    public void hello() {
         System.out.println("Hello from C3");
   }
}
```

Solution:

```
C3’s constructor is invoked
C2’s constructor is invoked
C1’s constructor is invoked
Hello from C1
C3’s constructor is invoked
C2’s constructor is invoked
Hello from C3
```

Exercise 9 (6 Marks)

The Java API defines the Comparable interface as

```
interface Comparable {
     int compareTo(Object);
}
```

where `x.compareTo(y)` should return -1 if `x` is less than y, 0 if they are equal, and 1 otherwise.

Define a class `MyArray` that implements `Comparable` and whose objects behave like integer arrays.

`MyArrays` should be compared based on the sum of their elements. For example,

```
int[] a = new int[] {1,2,3,4}; //create an array and initialize the elements
int[] b = new int[] {-1,2,-3,4,-5};
MyArray m1 = new MyArray(a); //the elements of m1 are those of a
MyArray m2 = new MyArray(b); //the elements of m1 are those of b
System.out.println(m1.compareTo(m2));
//prints 1, since 1+2+3+4 > -1+2-3+4-5
```

Define just enough of the `MyArray` class for the above code to work.

Solution:

```
class MyArray implements Comparable {
       private int[] a;

       MyArray(int[] b) { a = b; }
       public int compareTo(Object o) {
            MyArray other = (MyArray) o;
            int mysum = 0;
            int othersum = 0;
            for(int i=0; i<a.length;i++)
               mysum = mysum + a[i];
            for(int i=0; i<other.a.length;i++)
               othersum = othersum + other.a[i];
            if (mysum < othersum)
                return -1;
            else if (mysum == othersum)
                      return 0;
                 else return 1;
       }
}
```

Exercise 10 (6 Marks)

Given the following `GeometricObject` class:

```
public abstract class GeometricObject {
  private String color = "white";
  private boolean filled;
  /** Default constructor */
  protected GeometricObject() {
  }
  /** Construct a geometric object */
  protected GeometricObject(String color, boolean filled) {
    this.color = color;
    this.filled = filled;
  }
  /** Return color */
  public String getColor() {
    return color;
  }
  /** Set a new color */
  public void setColor(String color) {
    this.color = color;
  }
/** Return filled. Since filled is boolean, so, the get method name is isFilled */
  public boolean isFilled() {
    return filled;
  }
  /** Set a new filled */
  public void setFilled(boolean filled) {
    this.filled = filled;
  }

  /** Abstract method findArea */
  public abstract double findArea();

  /** Abstract method findPerimeter */
  public abstract double findPerimeter();
}
```

Create a new class called `Triangle` that extends the `GeometricObject` class to represent triangles. Your class should store the lengths of each side of the triangle.

The perimeter of a triangle is calculated by adding its sides.

The area of a triangle is calculated as follows:

$$s = \frac{(side1 + side2 + side3)}{2}$$

$$Area = \sqrt{s \times (s - side1) \times (s - side2) \times (s - side3)}$$

Solution:

```
public class Triangle extends GeometricObject {
    private double side1;
    private double side2;
    private double side3;
    public Triangle(double side1, double side2, double side3) {
        this.side1 = side1;
        this.side2 = side2;
        this.side3 = side3;
    }

    public double findPerimeter() {
        return side1 + side2 + side3;
    }

    public double findArea() {
        double s = (side1 + side2 + side3)/2;
        return Math.sqrt(s * (s-side1)*(s-side2)*(s-side3));
    }
}
```