German University in Cairo
Computer Science Department
Prof. Dr. Slim Abdennadher

June 12, 2006

Concepts of Programming Languages
Spring term 2006
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 15 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	∑
Possible Marks	10	10	8	8	8	15	6	7	8	15+5	100
Final Marks											

Exercise 1 (10 Marks)

a) What is the difference between strong and weak typing? Classify each programming language (Prolog, Haskell, C, Java) into the corresponding type.

b) What is the difference between unification and pattern-matching in Prolog?

c) What is Currying? Are all higher order functions curried?

d) What is meant by type inference in Haskell?

e) How can polymorphic types be restricted in Haskell? Give examples.

f) Why does the programmer have to perform manual garbage collection in C?

Exercise 2 (3+3+4=10 Marks)

For each of the following program fragments state the programming language and answer the questions.

a)
```
f n | n < 2 = [1]
    otherwise = f (n-1) ++ f (n-2)
```

- Language:

  Solution:
  Haskell
- Output of `f 5`:

  Solution:
  `[1,1,1,1,1,1,1,1]`

b) `public int f (int n) { return n<2 ? 1 : f(n-1) + f(n-2);}`

- Language:

  Solution:
  Java
- Output of `f(5)`:

  Solution:
  `8`

c) `older(X,Y) :- parent(X,Y); parent(X,Z), older(Z,Y).`

- Language:

  Solution:
  Prolog
- Purpose of the fragment:

  Solution:
  Make transitive relation based on `parent()`.
- Two facts that make `older(a,b)` true?

  Solution:

  ```
  parent(a,c).
  parent(c,b).
  ```

Exercise 3 (8 Marks)

Write a Prolog predicate `comm/5` that relates five **sorted** lists.

```
comm(Xs, Ys, OnlyXs, OnlyYs, Both)
```

holds when `OnlyXs` is a list of all elements of `Xs` that do not appear in `Ys`, `OnlyYs` is a list of all elements of `Ys` that do not appear in `Xs`, and `Both` is a list of the elements that appear in both lists.

Your code should work on `lists` of any sort of term, not just list of numbers, and **should traverse the list only once**.

```
?- comm([1,2,3],[2,4,5],OnlyXs, OnlyYs, Both).
OnlyXs = [1,3]
OnlyYs = [4,5]
Both = [2]
```

Solution:

Exercise 4 (6+2=8 Marks)

Given the following datatype

```
data Tree = Empty
          | Leaf Int
          | Node Tree Tree
```

a) Write a Haskell function `depth` that takes a tree and returns the depth of a tree.

b) Give the type of the function.

Solution:

```
depth :: Tree -> Int
depth Empty       = 0
depth (Leaf n)    = 1
depth (Node l r) = 1 + max (depth l) (depth r)
```

Exercise 5 (6+2=8 Marks)

a) Write a Haskell function `pairwise` that accepts a function `f` and a list of an even number of arguments. The function `pairwise` applies `f` to successive pairs of arguments and returns the list of results; an empty list should be returned if the number of arguments is odd. For example:

```
> pairwise + [1,2,3]
[]


> pairwise + [1,2,3,4,5,6]
[3,7,11]
```

Solution:

```
pairwise f l = if length l 'mod' 2 == 0 then
                   pairwisehelper f l else []

pairwisehelper f [] = []

pairwisehelper f (x:y:rest) = f x y : pairwisehelper f rest
```

b) Give the type of the function.

Solution:

```
pairwise :: (a -> a -> b) -> [a] -> [b]
```

Exercise 6 (3+3+3+6=15 Marks)

a) What is call by reference and how can C and Java perform the call by reference?

Solution:

C can simulate the call by reference by using pointers.

b) What does the following C program print?

```
main()
  {
    int x = 998;
    int y = 998;

    f(x,&y);
    print("%d %d\n", x, y);
   }

void f(int x, int *p)
  {
    x += 2;
    *p += 2;
  }
```

Solution:

```
998 1000
```

c) What does the following C program print?

```
main()
  {
   char alpha[] = {'A', 'B', 'C', 'D', 'E'};
   char x, *p1, * p2;

   p1 = alpha;
   p2 = p1 + 2;
   x = *p2;
   printf("%c%c%c", x, *p1, *(p2-1));
   }
```

Solution:

```
CAB
```

d) Write a C program that reverses a String in place, i.e. the reversed string should be the input string. **Do not create a new string**.

Solution:

```
main()
 {
   char str[101]
   scanf("%s", str);
   rev(str);
   printf("%s", str);
 }


void rev(char *s)
  {
   char t, *e;
   e = s + strlen(s) - 1;
   While (s < e)
    {
      t = *s;
      *s++ = *e;
      *e-- = t;
    }
}
```

Exercise 7 (3+3=6 Marks)

a) Consider the following code and write down its output. Assume that `a,b,aPtr,bPtr` are located at addresses `1000`, `2000`, `3000`, and `4000` respectively.

```
#include <stdio.h>

void main(){

   int a=2, *aPtr=&a, b=3, *bPtr=&b;

   printf("%u\n", &a);
   printf("%u\n", &bPtr);
   fun(aPtr, b);
   printf("%d\n", a);
   printf("%d\n", b);
}

void fun(int *x, int y){

  *x=*x*y++;
  printf("%u\n", x);
  printf("%d\n", y);
}
```

Solution:

```
1000
4000
1000
4
6
3
```

b) Consider the following code and write down its output. Assume that `x,y,xPtr`, and `yPtr` are located at addresses `8702, 9702, 8752`, and `9752` respectively. Assume that one integer length = 2 bytes.

```
#include <stdio.h>

void main(){

   int x=200, *xPtr=&x, y=300, *yPtr=&y;

   printf("%u\n", &y);
   fc(*xPtr, yPtr);
   printf("%d\n", x);
   printf("%d\n", y);
}

void fc(int a, int *b){

   a=a+1;
   b++;
   printf("%d\n", a);
   printf("%u\n", b);
}
```

Solution:

```
9702
201
9704
200
300
```

Exercise 8 (7 Marks)

Consider the following two classes

```
class B {
  public int i = 9;
  int j;
  private int k = 8;
  public void printMe(){
     System.out.println("i:" + i + "j:" + j + "k:" + k);
  }
  public B(int j) {
    printMe();
    this.j = j;
  }
}
```

```
class C extends B {
   int j;
   public C(int value) {
      super(value);
      i = 6;
      j = 10;
      printMe();
   }
}
```

Can the program compile? If not, why? If it can, what is printed when a new `C` object is created with `4` as argument to the constructor?

Solution:

The program can compile. The initializers in the super class gives `i` and `k` the values `9` and `8` respectively. The constructor in `B` is executed first, which print out:

```
i:9j:0k:8
```

then j is assigned the value `4`. The program returns to the constructor in the subclass. Here the `i` field in `B` is given the value `6`. The field `j` in `C` is given the value `10`. Then `printMe()` prints the fields from `B`:

```
i:6j:4k:8
```

Exercise 9 (3+5=8 Marks)

a) The `transfer()` method of the superclass `Account` calls both `deposit()` and `withdraw()` methods as shown below. Assume that `withdraw()` and `deposit()` methods are overridden in the subclasses `SAccount` and `CAccount`.

```
public boolean transfer(Account account, double amount) {
      if (withdraw(amount)) {
           account.deposit(balance);
           return true;
}
       else return false;
   }
```

From which classes the `withdraw()` and the `deposit()` methods will be called as a result of the statements below? Why?

```
SAccount acc1 = new SAccount("s12345","Slim",1000);
CAccount acc2= new CAccount("n12345","Noha",2000);
acc1.transfer(acc2,300);
acc2.transfer(acc1,100);
```

Solution:

```
SAccount acc1 = new SAccount("s12345","Slim",1000);
CAccount acc2= new CAccount(n12345","Noha",2000);
acc1.transfer(acc2,300); // withdraw of the SAccount,  deposit of the CAccount
acc2.transfer(acc1,100); // withdraw of the CAccount,  deposit of the SAccount
```

Dynamic Biding – the method associated with the object is being called.

b) Given that `Salesman` and `Clerk` are subclasses of `Employee` and that all classes have default constructors what will be the result of the following? (compilation error, run-time error, no problem)

- ```
  Employee e = new Salesman();
  Salesman s = (Salesman) e;
  ```

  Solution:
  No problem
- ```
  Employee e = new Salesman();
  Salesman s = e;
  ```

  Solution:
  Compilation error
- ```
  Employee e1 = new Salesman();
  Employee e2 = e1;
  ```

  Solution:
  No problem
- ```
  Employee e  = new Salesman();
  Clerk c = (Clerk) e;
  ```

  Solution:
  Runtime error
- ```
  Clerk c = new Salesman();
  Salesman s = (Salesman) c;
  ```

  Solution:
  Compilation error
  ```

Exercise 10 (15+5 Bonus Marks)

An organization has three types of employees, salesmen, hourly-rate employees and fixed income earners. For hourly rated employees salary is computed on hourly rate and the number of hours. If the number of hours exceed 150 (per month) they are paid a premium rate (1.5 times). For salesmen monthly income is 10% of the sales for that month. Fixed income earners have a fixed component and a productivity bonus.

a) Given that you are required to keep all employee objects in an array of superclass (Employee) references and compute the salary in a polymorphic way suggest one **abstract** method for the superclass. Provide a possible implementation for this method in all the subclasses (state your assumptions).

   Implement a complete Java program that consists of

   - one super class `Employee` and
   - three subclasses for `Salesman`, `HourlyRated` and `FixedIncome` with the following constructors:

     ```
     Salesman(String name, String taxID, double sales)
     Worker(String name, String taxID, double rate, double hours)
     FixedIncome(String name, String taxID, double amountt, double bonus)
     ```

b) To test your program and to show how the salaries can be computed in a polymorphic way write a class with a main method that initializes an array consisting of the following four employees and calculates the salary of these employees (using a loop):

   ```
   Salesman:     Bill Jones   d12345    120000.0
   Hourly rated: Timothy      d23668    12.0       180
   Salesman:     Mark Cheng   f13456    60000.0
   Fixed income: John Kennedy f23456    3000.0     250.0
   ```

   For a nice object-oriented implementation there will be 5 Bonus marks.

Extra Sheet

Extra Sheet

Extra Sheet