Spring 2016 Programming Languages Qualifying Exam

This is a closed book test. Clear, correct and concise responses will receive the best mark.

Correct, clear and precise answers receive full marks

Please start a new page for each question.

1. ***Parameter Passing (20 pts)***

List 5 different parameter passing methods. Indicate how each method is implemented and identify the challenges for each method. Please consider issues like performance, referential transparency, etc.

1) Value – values are copied from current context to new activation record Large data sets take time to copy can be slowed. No issues with referential transparency
2) Result –- resulting values are copied from called activation record to current context. Big challenges occur for something like prototype f(X,Y) is called as f(X,X) and f sets X and Y to different values. Performance issue if return value is large.
3) Value/Result -- combination of #1 and #2
4) Reference – memory references, so the called function effectively changes memory values in the caller context. This causes aliasing and hence the caller may not know that the memory is being changed. This has referential transparency issues.
5) Name – The symbol table needs to be maintained for this type of referencing to work. In this case the name of a variable is passed in. The memory location associated with the name is not determined until the actual instruction is evaluated. For example

void function f(X)
{ X =1 }

called with

f(Y)

will result in
Y=1, and the "Y" has to be in the scope of function f() either static scope or dynamic scope. The memory cell is the one related to "Y" in that scope. This can cause referential transparency issues because the name of the parameter is used instead of the value.

2. **Inheritance (20 pts)**
   There are 4 classes named A, B, C and D. These classes may or may not depend on each other. The classes have the following public methods respectively:
   - A has public method A.x()
   - B has public method A.x() and B.y()
   - C has public method A.x() and C.z()
   - D has public methods A.x(), B.y(), C.z()

a) Create a dependency diagram for these classes
b) Write a reasonable set of Java like declarations of the classes using the directive "extends". Each method has return type **void.**
c) Provide reasonable inheritance implementations.

a) **D depends on C and B, B depends on A, C depends on A**
b) **Class A { public void x() ;}**
   Class B extends A { public void y();}
   Class C extends A { public void z();}
   Class D extends B and C {}
c) **This example is the classical diamond problem. We have conflict between choosing method x() from B or C. Several solutions exist. 1) Make the user explicitly choose which method to use, 2) using order in how the object Class is created forces the preference, 3) only allow multiple inheritance on prototypes (generic) and not specific implementation.**

3. **Regular Expressions (15 pts)**

You are working on loading payroll data into a database. Each input line indicates a person, their pay over several pay periods, followed by the employee email address, and ending with the supervisor's last name. The file has the following format for each line:

FIRSTNAME, LASTNAME, PAY1, PAY2, PAY3,…,PAYN, EMAIL, SLASTNAME

1. FIRSTNAME, LASTNAME, and SLASTNAME are non-empty strings not containing comma ","

2. EMAIL is any string which contains exactly one ‘@’ and cannot contain a comma ‘,’ .
3. The PAY section has the following requirements:
    a. Each PAY element has the form of a decimal. A decimal is a string with at least one digit followed by a decimal point followed by exactly two digits. Examples include 10.30, 100.50, 0.00, and 9.52;
    b. There is at least one PAY element;
    c. Each PAY element is separated by a comma

Input examples include:

George Mike, Smith,0.20,100.40,Smith@homebound.com, Martinez
John,Smith,0.00,@,K
5908j85,jkf8044,0.00,0.12,3499.12,984.02, jkf8044@ksdkfj.klsjfjkl,5908j85

Create a regular expression which matches the above specification

^[^,]+,[^,]+,([0-9]+\.[0-9][0-9],)+[^,@]*@[^,@]*,[^,]+$

4. **Lazy Evaluation (30 pts)**

We wish to implement a form of infinite lists in Java or C++. We know from functional languages we can create a lazy data structure which creates elements on demand (random number generator is an example). For this problem, consider the infinite sequence of perfect positive squares, [1,4,9,16,25,….]. We wish to implement a new class called InfSqList (Infinite Squared List), which has the following properties

1. The object must maintain the least number of elements in a private list based on demand, and must keep track of this list for future use to improve performance,
2. The object implements the method **head()** which returns the first element from the list,
3. The object implements the method **tail()** which removes the first element from the list,
4. The object implements the method **nth(N)** which returns the Nth element from the list, where 0 is the smallest allowable value and maps to the first element in the list. head() and nth(0) are equivalent.

Your object must maintain an internal list which is filled with just enough elements to satisfy any query. For example, at the creation of the object, there are no internal values stored in the local list.

head(); // returns 1, and the list is now [1]
nth(5); // the list now has elements [1,4,9,16,25,36] and returns 36
nth(3); // the list is still [1,4,9,16,25,36] and returns 16
tail(); //returns nothing but makes the list [4,9,16,25,36]
nth(3);// returns 25
nth(5);// makes the list [4,9,16,25,36,49] and returns 49

```
import java.util.ArrayList;
import java.util.List;

/*  Class which implements an infinite sequence of perfect squares */
class InSqList {
       static  List L = new ArrayList();  // maintain list of generated
elements
       static int last=0;   // the last element we created

    public static void main(String[] args) {

     }// of main

     // Find the Nth element, if there are not enough, have produce
create more
     public static  int nth(int n) {
        if (L.size() < n+1  )
          produce(n-L.size() +2);

       return((int)L.get(n));
      } // of nth


// get the start of the list

      public static  int head() {
         if (L.size() == 0 )
           produce(1); // we need one, so produce it

        return((int)L.get(0));
       } // of head

      // Shorten the list if there is something
      public static void tail() {
         if (L.size() > 0)
           L.remove(L.get(0));// we know the list has unique elements
        }

      // put  N elements onto the end of the list.
      private static void  produce  (int n){ // create n more elements

         for (int i=1; i<= n; i++)
             { last++;
              L.add(last*last);  //add the square to the list
             }
      }// end of produce
} // of InSqList
```

5. **LISP – Functional Programming (15 pts)**

Write a LISP function (or set of functions) which when given two parameters, returns the element (or sub-matrix) of a multidimensional matrix represented as list of lists. For example

(select A L) Where A is a multidimensional array and L is the subscript string into A. A[0] would be the first element

(select '(1 2 3 4) '(2)) => 3 (read this as A[2] = 3)
(select '((1 2 3) (4 5 6) (7 8 9)) ' (1 2)) = 6 (read this as A[1][2] = 6)

The below functions perform the required actions
Function "select" moves across the reference list
Function "extract" pulls out the required stripe from the matrix input

```
(define (select A L )
  (if (null? L)
      A  ;; nothing to extract
      (select (extract A (car L))
              (cdr L))))


(define (extract A X) ;; single row extract
     (if (= X 0 )
        (car A)
        (extract (cdr A) (- X 1))))
```