# Comparative Programming Languages

# UCT CSC304 – Final Exam – January 2004

Question 1 is compulsory. Then answer either question 2 OR 3.

## Question 1 – General Concepts [15]

1.1. "Java is a universal programming language". State whether or not you agree, and provide a suitable reason. [2] (universal means we do not need any other languages)

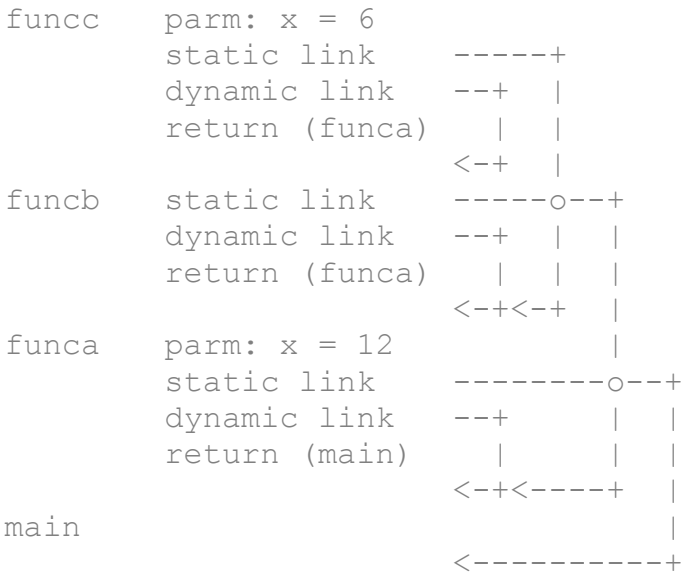
No. You cannot write a low-level operating system in Java.

1.2. To improve the orthogonality of user-defined data types, C++ allows the overloading of operators. List one advantage and one disadvantage of this feature. [3]

adv: It allows for more natural syntax in some domains, e.g., vector and matrix operations.
disad: It decreases readability since the essential meaning of an operator can be changed.

1.3. Draw the stack of activation records corresponding to the following ALGOL-like program when it is at "breakpointX". [6] (Assume static chains. Include all parameters, static and dynamic links).

```
program main ()
   subprogram funca ( int x )
   {
      subprogram funcb ()
      {
         subprogram funcc ( int x )
         {
            // breakpointX
         }
         funcc (6);
      }
      funcb ();
   }
   funca (12);
}
```




1.4. What is the value of the variable "c" after execution of the code below if the parameter is a) pass-by-value b) pass-by-name ? [4]

```
a = 1;
b = 2;
c = 0;
subprogram xyz ( integer x )
{
   a = 5;
   c = 30 + x;
}
xyz (a+b);
```

a) 33 b) 37

## Question 2 – Data Types [10]

2.1. Java does not have a primitive List data type – instead this functionality is supported by the class library. Explain the effect of this design decision on simplicity of the language. [2]

Simplicity of the language is increased since it doesn't need as many fundamental constructs.

2.2. Assume the following function is used to calculate the address of an element in an array with logical definition X[AS..AE][BS..BE] (where AS, BS represent lower bounds and AE, BE represent upper bounds):

```
Address X[i][j] = Address X[AS][BS] + [(i-AS) + (j-BS)] * n
  where n = size of element
```

2.2.1. How much memory is covered by the addresses computed for this array? [2]
2.2.2. If the problem domain guarantees that there will never be two combinations of indices i and j such that their sums are the same, the above function saves memory. What is its other advantage over the traditional matrix-like access function? [2]

2.2.1. (AE+BE+1)*n

2.2.2. it is faster since there are no multiplication operations in the simpler access function.

2.3. It is often said that Java's garbage collection "works worst when you need it most". Justify this statement by explaining under what conditions this occurs and why its performance is considered "worst". [4]

Garbage collection happens when there is no memory available – thus "need it most". The mark-and-sweep operation has to process all the memory (during its initialise and free operations) since all is in use when you run out – thus performance is "worst".

## Question 3 – Subprograms and Scope [10]

3.1. Dynamically-scoped languages are not very popular. State one reason for this. [2]

It is not easy to understand the effect of a subprogram out of context of its execution environment.

3.2. In an object-oriented programming language, when do instance variables (i.e., members of an instance of a class) have lifetime and when do they have scope? [4]

lifetime: whenever the instance they are part of has lifetime
scope: in all methods of the class, all methods of subclasses (when they are protected or public) and wherever the instance variable has scope (when they are public). also, in friend methods/classes.

3.3. While simple variable identifiers cannot have scope without lifetime, it is possible for an lvalue (an expression that can appear on the LHS of an assignment) to have scope without lifetime. Provide an example of such an expression and discuss. [4]

```
int *p;
*p = 12;
```

This results in an invalid memory access since the variable that the pointer points to (*p) has scope but not lifetime.