CS2104 – Programming Languages
School of Computing
National University of Singapore

August 25, 2010

Problem Set 1

Semester 1, 2011/12

Due: ***September 8, 23:59***

5 marks

Submission: In IVLE, in the cs2104 workbin, you will find a folder called `“Homework submissions”.` In that folder, there are currently ***3 subfolders:*** **`PS1P01,…,PS1P03`**. The last two digits of the folder name indicate the solution that is to be submitted into that folder: the solution to *Question 1* into **`PS1P01`**, and so on (that is, you need to submit 3 separate solutions to 3 questions). A solution should consist of a ***single text file*** that can be compiled, or loaded into the interpreter of interest and executed. You should provide as much supplementary information about your solution as you can, ***in the form of program comments***.

Problem 1 **[1 mark, submit to `PS1P01`]**

Write a VAL program that converts a string made up of digit characters into the unsigned numeric value corresponding to that string. Your program should assume that the sting resides in memory at the address pointed to by register **`esi`**, and should return the result in register **`eax`**.

Problem 2 **[2 marks, submit to PS1P02]**

Write an interpreter for the following language, which we shall call *L*. Programs of the language *L* are sequences of simple instructions (no whitespaces are allowed). Each instruction has the form

$$variable \otimes = operand;$$

where *variable* is a variable name, following the syntax of C identifiers, $\otimes$ is any of the operators +, -, *, or /, and *operand* is either a variable, or an unsigned integer. The instruction is terminated by semicolon. All the variables are considered to be initialized with 0 before the program starts execution. The interpreter should print the values of all the variables in the program at the end of the execution. Your interpreter should be implemented in C.

Execution example:

For the input string: “**`x+=3;y+=5;x*=y;`**”
Interpreter output: **`x = 15  y = 5`**
Hint: Implement your interpreter as a deterministic finite automaton.

Problem 3 **[2 marks, submit to PS1P03]**

In VAL, a linked list is implemented in the following way. Each list element is an 8-byte entity, whereby the first 4 bytes represent the data field of the element, and the second group of 4 bytes represent the pointer to the next element. In our case, the 4 data field bytes shall be interpreted as an integer. Thus, list elements are sequenced together by interpreting the second group of 4 bytes in an element as the address of the next element in the list. The last element in the list has this “next element” field set to 0 (implying that no list element can reside at address 0).

Write a VAL program that receives the address of a list in register **`esi`**, and **computes the sum of the elements in the list**. To test your program, write a **`main`** function that sets up a linked list with the elements placed in memory in as much a random fashion as possible. Your **`main`** function should then call **`exec()`**, and then finally print the value of **`eax`**, which should be the desired result of the VAL program.

Further Practice Problems

These problems are for your own individual practice. Solutions are not to be submitted, and will not be marked. You are, however, allowed to post your solutions in the forum for comparison and discussion. Good posts will earn marks.

Further Practice Problem 1

Considering the encoding of a tree given in PS1P02, write a procedure that computes the height of the tree. The height is defined in the following way: the height of a leaf is 0, and the height of a non-leaf tree is one plus the maximum of the heights of the tree's children.

Further Practice Problem 2

Considering the encoding of a tree given in PS1P02, write a procedure that takes in a list of integers, and returns the heap containing the elements in the list (a heap is a tree whereby the root of each subtree is the minimum of all the elements in that subtree).

Further Practice Problem 3

Write an Oz procedure that returns the list of the first N prime numbers, where N is a parameter of the procedure.

Further Practice Problem 4

Write an Oz procedure that computes the prefx sums of a list of integers. In the list of prefix sums, the `N`th element is the sum of the first `N` elements of the original list.

Further Practice Problem 5

Write an Oz procedure that takes in a list of integers, and returns `true` if the numbers in the list are consecutive, and `false` otherwise.