2013 Summer Entrance Examination

Department of Creative Informatics
Graduate School of Information Science and Technology
The University of Tokyo

Programming

INSTRUCTIONS

1. Do not open this problem brochure until the signal to begin is given.
2. Write your examinee ID below on this cover.
3. An answer sheet and a draft sheet accompany this brochure. Write down your examinee ID on these sheets.
4. The USB memory delivered beforehand to each examinee contains ASCII text files `prog1.txt`, `prog2.txt`, `prog3.txt`, `prog4.txt`, and `prog5.txt`.

   Before the examination starts, copy these files to your PC and browse them. Verify that you can see the text files and then remove your hands from your PC. If you cannot read the files properly, consult the test supervisor. The contents of the USB memory are common to all examinees.
5. You may choose your favorite programming languages.
6. You may consult only one printed manual of a programming language during the examination. *You can use or copy any libraries or program segments existing in your PC, but you cannot connect to the Internet.*
7. By the end of the examination, make a directory/folder on your PC, whose name is the same as your examinee ID, and put your program files and related files into the directory/folder. Copy the directory/folder onto the delivered USB memory.
8. At the end of the examination, the USB memory, the answer sheet and the draft sheet are collected.
9. After these are collected, stay at your seat, until all examinee program results have been checked briefly by the test supervisor.
10. After the brief check, try to save your program execution environment on the PC so that you can run your program as soon as possible during the oral examination in the afternoon.
11. *Leave your PC and this brochure together in the room* for the oral examination and leave the room until you are called.

Examinee ID _____

This page is blank.

This page is blank.

L is a very simple programming language. It has only five instructions shown in Table 1. Only `x` and `y` are variables available in L. For example,

```
SET x 1
SET y -2
ADD x y
PRN x y
```

This code sets a variable `x` to `1` and sets `y` to `-2`. Then it computes `x + y` and stores the result in `y`. Finally, it prints the values of `x` and `y` and terminates.

Table 1: L's instructions

`ADD` α β	Add α to β. The new value of β is $\alpha + \beta$. β must be a variable name.
`CMP` α β	Skip the next instruction if α equals β. Otherwise, move to the next instruction.
`JMP` α β	Jump to the instruction α lines below the current one (If α is a negative integer, then jump backward). β is not used.
`PRN` α β	Print α and β and then terminate the execution.
`SET` α β	Set α to β. α must be a variable name.

α and β are either an integer literal or a variable name.

(1) The USB memory stick contains `prog1.txt`, which is a code written in L. Write a program that reads this L code and prints the first operand of each line. For example, if a line is:

```
SET x 1
```

then the first operand is `x` and the second is `1`. So `x` is printed.

(2) Explain the behavior of the following L code:

```
SET x 1
SET y 0
ADD x y
ADD 1 x
CMP x 10
JMP -3 0
PRN x y
```

(3) Write a program that reads an L code and executes it. Test your program with `prog2.txt` on the USB memory stick. You do not have to implement the instructions `CMP` or `JMP`, which are not used in `prog2.txt`. You can assume that only valid L code is given to your program and the size of the L code is less than 100 lines.

(4) Extend your program written for (3) to support all the instructions in Table 1. Make any words consisting of `a` to `z` available as variables in L. For example, `i` and `count` are variables. Test your program with `prog3.txt` on the USB memory stick.

(5) Extend your program written for (4) to support the new instructions `SUB` (subroutine call) and `BAK` (go back) in Table 2. Allow nested subroutine calls. Test your program with `prog4.txt` on the USB memory stick.

(6) Extend your program written for (5) to support the new instructions `CAL` (call) and `RET` (return) in Table 3. These instructions are used for a function call. Test your program with `prog5.txt` on the USB memory stick. Note that variables after CAL till RET must be treated as local variables. The instructions must allow recursive function calls. A function argument is available through a special variable `in` till `RET` is executed while a return value of the last function call is available through a special variable `out`.

Table 2: `SUB` and `BAK`

`SUB` α β	Record a return position and jump to the instruction α lines below the current one (if α is a negative integer, jump backward). β is not used.
`BAK` α β	Return to the position recorded by `SUB`. α and β are not used.

α and β are either an integer literal or a variable name.

Table 3: `CAL` and `RET`

`CAL` α β	Record a return position and jump to the instruction α lines below the current one (if α is a negative integer, jump backward). β is an argument.
`RET` α β	Return to the position recorded by `CAL`. α is a return value. β is not used.

α and β are either an integer literal or a variable name.

This page is blank.

This page is blank.