Homework #1

TA in charge: Sihyun Kim
E-mail: sihyun.kim@kaist.ac.kr

I. Goal of this assignment

- ✓ Understanding how computers work at a low level, bridging the gap between high-level programming languages and the hardware.
- ✓ Implementing procedures with recursive calls in the MIPS ISA.

II. What to implement

- ✓ You need to implement a program that constructs a Binary Search Tree (BST) and performs a postorder traversal on it.
  - A BST is a binary tree that satisfies the following conditions:
  - All keys in the left subtree are smaller than the key of the node.
  - All keys in the right subtree are larger than the key of the node.
  - Both the left and right subtrees are also BSTs.
- ✓ Here is the input format you should follow:
  - The first line contains the number of nodes N.
  - Each of the next N lines contains the key of a node
- ✓ Here is the output format you should follow:
  - Print the result of postorder traversal of the BST, with each key on a separate line.

III. Constraints

- ✓ The number of nodes N: $1 \leq N \leq 1{,}000$
- ✓ The key of each node K: $1 \leq K \leq 2{,}147{,}483{,}647$
- ✓ All node keys are unique (no two nodes share the same key)
- ✓ We will use the SPIM simulator to check if your assembly program runs correctly.
- ✓ *Your program **must be runnable on the simulator**. Otherwise, you will not get any points except for the report. Please check your program before submission.*

IV. Example of input and output

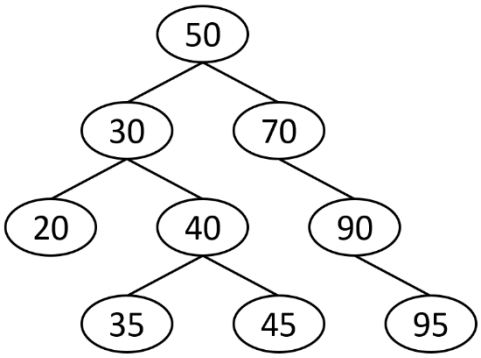
■ Input

```
9
50
70
30
40
20
90
45
35
95
```

■ Output

```
20
35
45
40
30
95
90
70
50
```

■ BST



V. Submission and grading

✓ Your submission should include: **(Total 100 pts)**

A. Source code file of your assembly program [hw1_StudentID.s]
The source code should contain:

i. Implementation of the given algorithm in assembly language **(20 pts)**

ii. Comments explaining the details of your implementation **(10 pts)**

iii. Code that correctly receives integer input and prints BST traversal results to the simulator console

We will check whether the correct results are produced for test cases. **(40 pts)**

B. Brief report [hw1_StudentID.pdf]
The content of the report should describe:

i. Stack allocation layout for each procedure **(10 pts)**
Describe the content of the stack space for each procedure; show the position of each data in the stack space using an offset to the stack pointer of each procedure.

ii. Brief explanation of your implementation **(20 pts)**
Describe what you have considered for your implementation, implementation issues, etc.

There is no specific format for the report. You should submit your report in either English or Korean.

- ✓ **Upload these two files to KLMS**.

VI. Due date

- ✓ **23:59, Oct. 10 (Fri.)**
- ✓ **Late** submission due date: 23:59, Oct. 11 (Sat.)
- ✓ For late submissions, there will be a 50% penalty on your total score.
- ✓ **You cannot submit after the late submission due date**.

VII. Cheating

- ✓ If there is any cheating in your submission, you will get 0 points.
- ✓ We will do a similarity check on the submitted code files to catch plagiarism between students.
- ✓ *The following will be regarded as cheating*:

A. Copying other students' simulation results or reports

B. Modifying other students' results and using them as if they were your own

C. Using other sources without any references

D. All other sorts of inappropriate behaviors.

VIII. Tips & Notes

- ✓ Since SPIM does not simulate a delay slot in the default configuration, you can ignore it for this assignment.
- ✓ The simulator may not be able to load your code if it contains non-English characters.
- ✓ You can use **pseudo-instructions** supported in SPIM for your convenience.
- ✓ You would need to review how to implement procedure calls with MIPS assembly.
- ✓ It would be helpful to first implement the assignment in the C language.
- ✓ **If you have little experience with assembly programming**, this assignment could take much longer than expected. If you think this is your case, we **recommend you begin this assignment as early as possible**.
- ✓ If you have any questions, please use the KLMS Q&A board.
- ✓ You can use SPIM(terminal version) to test your code. You can download it from https://sourceforge.net/p/spimsimulator/code/HEAD/tree/.

  Directory tree:

```
├── bin
├── CPU
|   └── exceptions.s
├── Documentation
├── Setup
├── spim
|   ├── Makefile
|   ├── spim
|   └── hw1_20250000.s
└── Tests
```

  Testing command:

```
make spim
./spim -ef ../CPU/exceptions.s -file hw1_20250000.s
```