2022 Fall Computer Architecture

Homework 2

Due date: 10/18 23:59

Description

In this homework, you are going to use Jupiter RISC-V simulator to implement the Fibonacci sequence and the inorder traversal of a complete binary tree.

After finishing this homework, you will be familiar with Jupiter basic I/O, RISC-V calling convention, and the implementation of array and pointer in assembly level.

Requirements

1. Fibonacci Sequence

Given an integer n, your program should output the n^{th} item of the Fibonacci sequence.

The definition of Fibonacci sequence is as follows.

$$F_0 = 0$$
$$F_1 = 1$$
$$F_n = F_{n-1} + F_{n-2}$$

Input format	Output format
n $(0 \leq n \leq 15)$	[Result of F_n]
Sample Input 1	Sample Output 1
0	0
Sample Input 2	Sample Output 2
2	1
Sample Input 3	Sample Output 3
5	5

2. Inorder traversal of a complete binary tree

Given a complete binary tree, your program should output the inorder traversal of such a tree.

A complete binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nodes in the last level are as far left as possible. Take Figure 1 as an example, tree (a) and tree (b) are complete binary trees, while tree (c) is not.

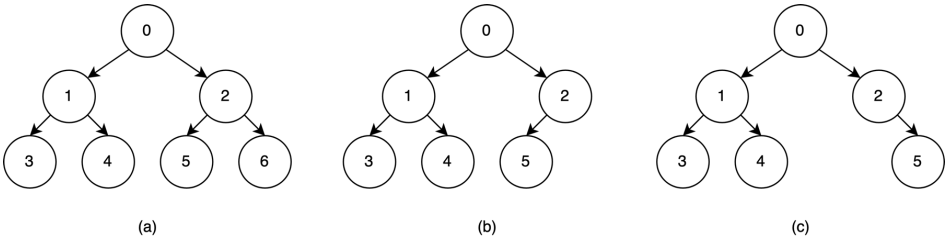


Figure 1. Complete Binary Tree

Input

The first line of input for each test case contains a single integer n, the number of nodes of the complete binary tree.

This is followed by n lines, each corresponding to a node value of the complete binary tree. These n node values are stored in an array $a_0 \ldots a_{n-1}$ in sequence to represent a complete binary tree. We define that given a parent node a_i, the left child node can be accessed using a_{i*2+1} and the right child node can be accessed using a_{i*2+2}. Figure 2 gives an example of the memory layout.

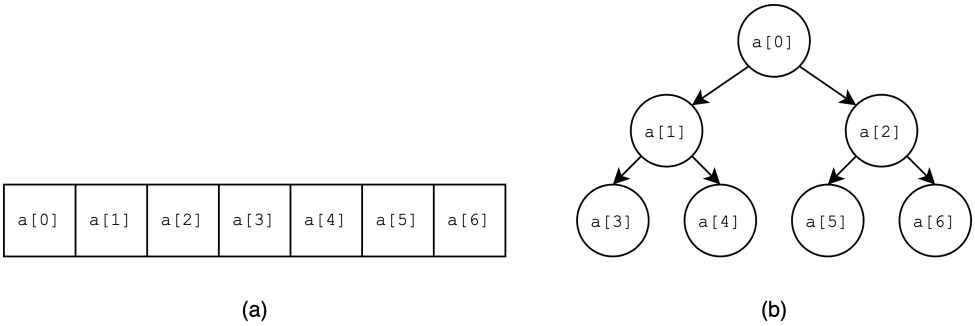


Figure 2. Memory layout of a complete binary tree

Output

The output should contain n numbers, each separated by a space. For simplicity, the trailing space is allowed. (1 2 3 and 1 2 3 are the same.)

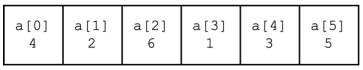
These n numbers are the node values of the complete binary tree, following by the order of inorder traversal.

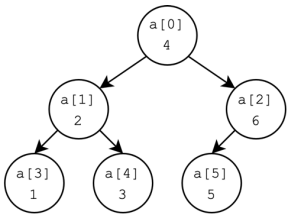
If n is equal to 0, your output should be empty.

Input format	Output format
$n\ (0 \leq n \leq 10{,}000)$ $a_0 (0 \leq a_i \leq 2{,}147{,}483{,}647)$ a_1 ... a_{n-1}	[Inorder traversal of $a_0 \ldots a_{n-1}$]

Sample Input 1	Sample Output 1
0	$< empty >$

Sample Input 2	Sample Output 2
1 3	3





Sample Input 3	Sample Output 3
6 4 2 6 1 3 5	1 2 3 4 5 6

Grading policy

We will judge the correctness of your program by running the following commands.

```
$ jupiter [student_id]_hw2_fibonacci.s < input_file

$ jupiter [student_id]_hw2_inorder.s < input_file
```

- There are 6 testcases for the Fibonacci sequence, 4 testcases for the inorder traversal of a complete binary tree, 10 points per testcase.
- Time limit: 60 seconds per testcase.
  - Time limit is only used for auto judgement. If you can output correct answers but can't meet the timing requirement, please contact TA using email.
  - However, it's very likely that your program has some bugs if it's stuck for 1 minutes. (infinite loop, stack become a mess…etc)
- 10 points off per day for late submission.
- You will get zero point if we find out that you solve the problem without using recursion.
- You will get zero point if we find out that you solve the problem by storing all possible answers and print it out directly.
- You will get zero point for plagiarism.

Submission

Due date: 10/18 23:59

You are required to submit **.zip** file to NTU Cool.

Please rename your program **[student_id]_hw2_fibonacci.s** for the Fibonacci sequence, **[student_id]_hw2_inorder.s** for the inorder traversal of a complete binary tree and pack 2 files using the following folder structure:

```
[student_id (lower-cased)].zip
    /[student_id]/ <-- folder
        [student_id]_hw2_fibonacci.s <-- file
        [student_id]_hw2_inorder.s <-- file
```

For example, if your student id is **b12345678**, your zip file should have followed structure:

```
b12345678.zip
    /b12345678/ <-- folder
        b12345678_hw2_fibonacci.s <-- file
        b12345678_hw2_inorder.s <-- file
```

Reference

- Lecture slides
- Jupiter RISC-V simulator
  https://github.com/andrescv/Jupiter
- Jupiter RISC-V simulator docs
  https://github.com/JupiterSim/Docs
- RISC-V Instruction Set Manual
  https://github.com/riscv/riscv-isa-manual
  https://riscv.org/technical/specifications