2023 Fall Computer Architecture

Homework 2

Due date: 10/16 23:59

Description

In this homework, you are going to use Jupiter RISC-V simulator to implement two recursive functions, one is to calculate the recurrence relation $T(n)$ and the other is to print out a linked list.

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. Recurrence Relation

Given an integer n, your program should calculate T(n).

The definition of recurrence relation is as follows.

$$T(n) = \begin{cases} 2 \times T(n-1) + T(n-2) & , if\ n \geq 2 \\ 1 & , else\ if\ n = 1 \\ 0 & , else\ if\ n = 0 \end{cases}$$

Input format	Output format
n $(0 \leq n \leq 15)$	[Result of T_n]
Sample Input 1	Sample Output 1
0	0
Sample Input 2	Sample Output 2
2	2
Sample Input 3	Sample Output 3
5	29

2. Print Out Linked List

Given a linked list, your program should traverse and print out all elements of the linked list reversely.

Linked list is a data structure that uses references to connect its elements instead of storing its elements in sequential memory. In addition, linked lists support efficient removal of elements compared to arrays, which would require shifting all the elements. Although finding a specific element in a linked list can take more time than in an array, linked list is still a useful data structure for certain applications, such as an LRU (Least Recently Used) cache.

Input

The first line of input for each test case contains a single integer n, the number of nodes of the linked list.

The following n lines, each corresponding to a value push front to the linked list.

Output

The output should print out all elements of the linked list reversely, each separated by a space. Take Figure 1 for example, if the linked list like the figure, you should print out 10,4,1,3. For simplicity, the trailing space is allowed. (1 2 3 and 1 2 3 are the same.)

If there are no elements in the linked list, your output should be empty.

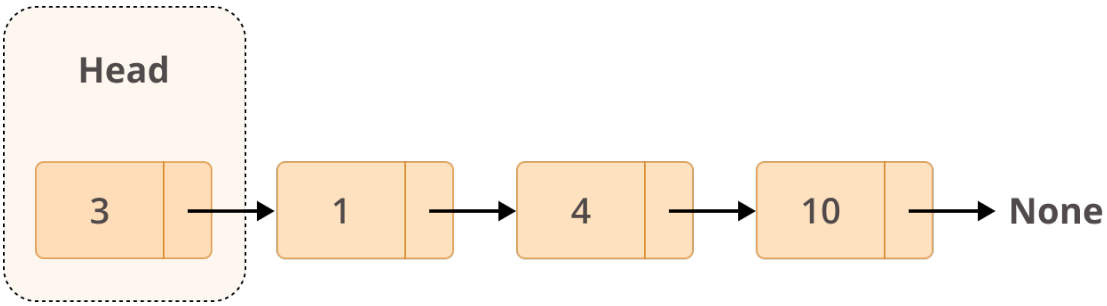


Figure 1. Linked List

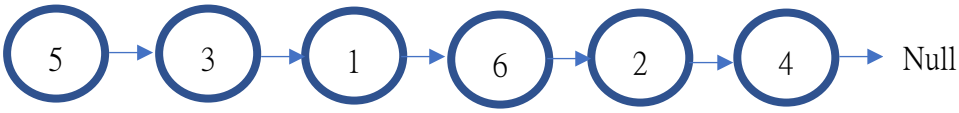
Input format	Output format
$n\ (0 \le n \le 10{,}000)$ $a_0 (0 \le a_i \le 2{,}147{,}483{,}647)$ a_1 ... a_{n-1}	$[a_0 a_1 \dots a_{n-2} a_{n-1}]$

Sample Input 1	Sample Output 1
0	$Empty$!

Sample Input 2	Sample Output 2
1 3	3

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

Linked list of the sample input 3 :



Grading policy

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

$ jupiter [student_id]_recurrence.s < input_file

$ jupiter [student_id]_linkedlist.s < input_file

- There are 6 testcases for the recurrence relation, 4 testcases for the traverse of the linked list, 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 for plagiarism.

Submission

Due date: 04/04 23:59

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

Please rename your program **[student_id]_recurrence.s** for the recurrence relation, **[student_id]_linkedlist.s** for the print out a linked list and pack 2 files using the following folder structure:

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

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

```
b12345678.zip
    /b12345678/ <-- folder
        b12345678_recurrence.s <-- file
        b12345678_linkedlist.s <-- file
```

Reference

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