# Lab 5 - Operations in a Linked list

The goal of this lab assignment is to understand how a data structure defined at high level (in Java) can be translated into MIPS language. You will also see how *syscalls* can be used to dynamically allocate memory in the MIPS simulator.

It goes without saying that all procedures you write in this assignment should work if called on other inputs besides the ones given in the starter code. Make sure your program run successfully and you double check your modified assembly file.

## Code skeleton for Linked Lists

The starter file listnode.asm (download here) contains a skeleton of a MIPS implementation for the Java class shown below.

```
class ListNode {
    int data;
    ListNode next;

    public ListNode(int data) {
        this.data = data;
        this.next = null;
    }

    public static int[] testvalues = { 2, 6, 3, 0 };

    public void append(int data) {
        if (this.next==null) {
            this.next = new ListNode(data);
        } else {
            this.next.append(data);
        }
    }

    public void printlist() {
        // your job!
    }

    public void remove(int data) {
        // your job!
    }

    public static void main(String[] args) {
        ListNode head = new ListNode(testvalues[0]);
        for (int i=1; i<testvalues.length; i++) {
            head.append(testvalues[i]);
        }

        head.printlist();
    }
}
```

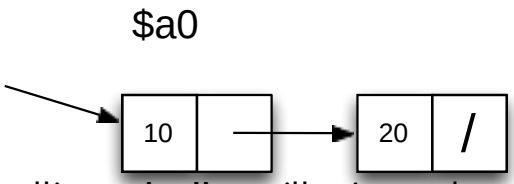
This ListNode class is like the one discussed in the lecture on data structures. Your job is to fill in the missing **printlist** and **remove** methods.

1) Assemble and run the code. Examine the code and the register values to find where the linked list is stored. Remember, it is a **linked list not an array**. The test program copies an array in the .data section starting at 0x10010000 to a linked list by calling the append method in a loop.

2) Write the printlist method. It takes one argument: the address of the first ListNode. It prints the list with values separated by commas.

3) Write the remove method. It takes two arguments: the address of the first ListNode and the data to be removed from the list. It updates the list by removing this data element. *You only need to remove the first occurrence in case there are multiple copies of the same value in the list.*

For example, if $a0 stores the address of the following list




Then calling `printlist` will print to the console:

```
10,20,
```

Note that the trailing comma (,) is expected. Requirements:

- printlist must use the standard MIPS procedure conventions that were shown in class
- printlist must be implemented using recursion

If it helps, you may assume that printlist is never called with the argument NULL (i.e., empty list).

## Hints:

- Just like we've provided the Java code for the constructor and append method, it may help you to write the Java version of `printlist` then translate the algorithm to MIPS.

- To print an integer to the terminal, use the "print integer" syscall. You can find information in MARS under Help > MIPS > Syscalls.

- To print an ascii character to the console use the "print character" syscall. Note that the MARS is nice to us, so if you put a character in single quotes (e.g., `','`) instead of an immediate then it will convert it to ascii encoding for you.