Doubly-linked List Based on Raw Pointer

beta version

```rust
// we need null *raw pointer*, ptr::null()
use std::ptr;

pub struct List {
    index: *mut ListItem,    // raw pointer
}

pub struct ListItem {
    value: u32,
    next: *mut ListItem,
    previous: *mut ListItem,
    owner: *mut List,
}

impl List {
    pub fn new(item: &mut ListItem) -> Self {
        List {
            index: item as *mut ListItem,
        }
    }
    pub fn append(&mut self, item: &mut ListItem) {
        // just a beat version, no need to implement all functions
        // raw pointer dereference should be written in unsafe block
        unsafe {
            (*self.index).next = item as *mut ListItem;
        }

    }
}

impl ListItem {
    pub fn new(value: u32) -> Self {
        ListItem {
            value: value,
            next: ptr::null_mut(),
            previous: ptr::null_mut(),
            owner: ptr::null_mut(),
        }
    }
}

#[cfg(test)]
mod test {
    use super::*;
    #[test]
    pub fn test_new() {
        let mut itemFirst: ListItem = ListItem::new(2333);
        assert_eq!(itemFirst.value, 2333);

        let mut list: List = List::new(&mut itemFirst);
        let mut itemSecond: ListItem = ListItem::new(9999);
        assert_eq!(itemSecond.value, 9999);
        list.append(&mut itemSecond);
        unsafe{
            assert_eq!((*(*list.index).next).value, 9999);
        }

    }
}


```